Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Parametrovo efektívna viacjazyčná adaptácia veľkých jazykových modelov prostredníctvom Extreme Gradient Boosting LoRA

Bakalárska práca

2025
Aleksandr Bukhtoiarov

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Parametrovo efektívna viacjazyčná adaptácia veľkých jazykových modelov prostredníctvom Extreme Gradient Boosting LoRA

Bakalárska práca

Študijný program:	Informatika
Študijný odbor:	Informatika
Školiace pracovisko:	Katedra informatiky
Školiteľ:	Mgr. Marek Šuppa

Bratislava, 2025
Aleksandr Bukhtoiarov

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Aleksandr Bukhtoiarov
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Parameter-Efficient Multilingual Adaptation of Large Language Models via Extreme Gradient Boosting LoRA
Parametrovo efektívna viacjazyčná adaptácia veľkých jazykových modelov prostredníctvom Extreme Gradient Boosting LoRA

Anotácia: Táto práca skúma rozšírenie adaptácie Extreme Gradient Boosting Low-Rank Adaptation (XGBLoRA) pre veľké jazykové modely (LLM) do viacjazyčných kontextov s osobitným zameraním na nedostatočne zastúpené jazyky. XGBLoRA, pôvodne navrhnutá na zlepšenie výpočtovej efektivity pri zachovaní výkonu, spája princípy Low-Rank Adaptation (LoRA) s Gradient Boosting. Metóda umožňuje parametricky efektívne jemné ladenie a má potenciál znížiť výpočtovú stopu a zlepšiť adaptabilitu modelov naprieč jazykmi.

Cieľ: Ciele bakalárskej práce zahŕňajú (ale nie sú obmedzené na)
- reimplementácia metódy Extreme Gradient Boosting Low-Rank Adaptation (XGBLoRA) a reprodukovanie publikovaných výsledkov na overenie jej výkonnostných nárokov v rôznych jazykových kontextoch.
- analyzovanie súčasných najmodernejších metód na parametricky efektívnu adaptáciu vo veľkých jazykových modeloch (LLM) so zameraním na teoretické a praktické dôsledky nízkoúrovňových úprav vo viacjazyčnom prostredí.
- aplikáciu XGBLoRA na dolaďovanie viacjazyčných LLM na slovenských súboroch údajov, systematické vyhodnocovanie výkonnosti oproti zavedeným metódam založeným na LoRA.
- analyzovanie výstupov a vzorcov chýb v rôznych jazykoch, identifikácia výziev, ktoré sú jedinečné pre dolaďovanie viacjazyčného LLM a oblasti, v ktorých XGBLoRA poskytuje výhodu v oblasti efektívnosti alebo výkonu.

Literatúra:
Zhang Y., et al. “Less is More: Extreme Gradient Boost Rank-1 Adaption for Efficient Finetuning of LLMs” 2024
Hu, Edward J., et al. "Lora: Low-rank adaptation of large language models." arXiv preprint arXiv:2106.09685 (2021).
Friedman, Jerome H. "Stochastic gradient boosting." Computational statistics & data analysis 38.4 (2002): 367-378.

Vedúci: Mgr. Marek Šuppa
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: doc. RNDr. Tatiana Jajcayová, PhD.

Abstrakt

Táto práca skúma rozšírenie adaptácie Extreme Gradient Boosting Low-Rank Adaptation (XGBLoRA) pre veľké jazykové modely (LLM) do viacjazyčných kontextov s osobitným zameraním na nedostatočne zastúpené jazyky. XGBLoRA, pôvodne navrhnutá na zlepšenie výpočtovej efektivity pri zachovaní výkonu, spája princípy Low-Rank Adaptation (LoRA) s Gradient Boosting. Metóda umožňuje parameterovo efektívne dolaďovanie a má potenciál znížiť výpočtovú stopu i rozšíriť adaptabilitu modelov naprieč jazykmi.

V rámci práce bola XGBLoRA nanovo implementovaná a prispôsobená na multilingválne prostredie. Navrhujú sa dve praktické modifikácie — zvýšenie ranku adaptérov nad pôvodne odporúčanú hodnotu $r = 1$ a SVD-inicializácia — ktoré zohľadňujú špecifiká jazykovej adaptácie. Experimentálna časť pokrýva anglický benchmark GLUE a slovenský dataset otázka-odpoveď SKQuAD; testované boli modely RoBERTa-base, GPT-Neo-125M a BERT-base-uncased.

Výsledky ukazujú, že XGBLoRA systémovo prekonáva klasickú LoRA pri rovnakom alebo menšom počte trénovaných parametrov: na GLUE zvyšuje priemernú presnosť z 79,7% na 81,3% (+1,6 p. b.) a na SKQuAD zvyšuje F1 skóre modelu GPT-Neo z 37,2% na 39,6% (+2,4 p. b.). Pri BERT-base bolo dosiahnuté F1 = 54,4%, čo prekonáva LoRA o približne 3 p. b. a zmenšuje odstup od plného doladenia na 8 p. b. Analýza chýb potvrdzuje, že postupné zvyšovanie ranku v štýle boostingu umožňuje modelu korekovať rozdielne typy chýb bez výrazného rizika pretrénovania.

Prínosom práce je (i) otvorená implementácia XGBLoRA kompatibilná s HuggingFace PEFT, (ii) systematické porovnanie rôznych PEFT stratégií pri adaptácii na slovenčinu a angličtinu a (iii) odporúčania pre voľbu ranku, počtu iterácií a inicializácie adaptérov. Výsledky demonštrujú, že XGBLoRA predstavuje praktický kompromis medzi kvalitou a výpočtovými nárokmi a že výrazne zlepšuje prenos LLM do jazykov s obmedzenými zdrojmi.

Abstract

This thesis investigates the extension of Extreme Gradient Boosting Low-Rank Adaptation (XGBLoRA) for Large Language Models (LLMs) into multilingual contexts, with a particular focus on underrepresented languages. XGBLoRA, originally designed to improve computational efficiency while retaining performance, merges the principles of Low-Rank Adaptation (LoRA) with Gradient Boosting. The method enables parameter-efficient fine-tuning, and has the potential to reduce the computational footprint and improve model adaptability across languages.

In this work, XGBLoRA was re-implemented and tailored for a multilingual environment. Two practical modifications are introduced: (1) raising the adapter rank beyond the originally recommended value of $r = 1$, and (2) an SVD-based initialization, both motivated by the specifics of language adaptation. Experiments span the English GLUE benchmark and the Slovak question-answering dataset SKQuAD, using RoBERTa-base, GPT-Neo-125M, and BERT-base-uncased.

Empirical results show that XGBLoRA consistently outperforms standard LoRA with an equal or smaller number of trainable parameters: on GLUE it lifts average accuracy from 79,7% to 81,3% (+1,6 pp), and on SKQuAD it raises GPT-Neo's F1 score from 37,2% to 39,6% (+2,4 pp). For BERT-base, an F1 of 54,4% surpasses LoRA by roughly 3 pp and narrows the gap to full fine-tuning to 8 pp. Error analysis confirms that the boosting-style, step-wise rank increase lets the model correct diverse error types with minimal risk of over-fitting.

The contributions are: (i) an open XGBLoRA implementation compatible with HuggingFace PEFT, (ii) a systematic comparison of PEFT strategies for English and Slovak adaptation, and (iii) practical guidelines on adapter rank, iteration count, and initialization. Overall, XGBLoRA provides a pragmatic trade-off between quality and computational requirements and markedly improves the transfer of LLMs to low-resource languages.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Kapitola 1

Úvod

Stručný obsah kapitoly. Táto kapitola predstavuje východiská a kontext práce, formuluje cieľ a konkrétne úlohy výskumu a zdôvodňuje aktuálnosť zvolenej témy. Diskutuje sa potreba parameterovo efektívnej adaptácie veľkých jazykových modelov a navrhované riešenie *XGBLoRA*. Na záver kapitoly je predstavená štruktúra celej bakalárskej práce.

1.1 Zadanie a kontext

Moderné veľké jazykové modely (LLM) obsahujú miliardy parametrov, čo sťažuje ich efektívne dolaďovanie na nové úlohy alebo jazyky. Klasické plné doladenie vyžaduje aktualizáciu všetkých váh a obrovské výpočtové zdroje. Preto narastá význam metód, ktoré dokážu model prispôsobiť pri oveľa menšom počte upravovaných parametrov.

Nízkorankova adaptácia LoRA (*Low-Rank Adaptation*) ponecháva pôvodné váhy nemenné a ku každému vybranému lineárnemu vrstveniu pridáva dve malé trénovateľné matice nízkeho ranku, ktorých súčin tvorí korekciu váh. Počet trénovaných parametrov tak klesá často pod 1 % a výkon zostáva porovnateľný s plným doladením.

Metóda *Extreme Gradient Boosting LoRA (XGBLoRA)* bola navrhnutá na prekonanie istých obmedzení základnej LoRA. Inšpiruje sa technikou gradientového boostingu: namiesto jednorazového natrénovania jedného LoRA modulu trénuje sekvenciu nízkorankových adaptérov, z ktorých každý cielene zlepšuje chyby predchádzajúcej iterácie. Takto postupne „zosilňuje“ model podobne ako ansámblové boostovacie metódy a zachováva výhodu malého počtu trénovateľných parametrov.

Pôvodná práca o XGBLoRA demonštrovala zlepšenie výkonu na anglických úlohách; cieľom tejto bakalárskej práce je rozšíriť a otestovať metódu v multilingválnom prostredí, konkrétne pre slovenský jazyk, ktorý je v korpusoch LLM zastúpený len okrajovo.

1.2 Aktuálnosť témy

S rastúcou veľkosťou modelov rastie aj praktická potreba efektívnych adaptačných techník. Objavilo sa viacero PEFT-prístupov (LoRA, AdaLoRA, FanLoRA, LoRA-XS, OLoRA), ktoré ukazujú, že so zlomkom parametrov možno dosiahnuť výkon blízky plnému doladeniu. Stále však existuje priestor na zlepšenie-najmä pri kombinácii viacerých adaptácií a pri adaptácii na menej zastúpené jazyky.

Slovenčina patrí medzi jazyky s obmedzenou podporou v LLM; schopnosť rýchlo a parameterovo efektívne adaptovať model na slovenské dáta má priame praktické využitie (lokalizované inteligentné asistenty, automatická analýza textu, atď.). Metóda XGBLoRA sľubuje vyššiu kvalitu adaptácie pri minimálnom navýšení parametrov-čo môže byť kľúčové v multi-užívateľských alebo multi-jazykových scenároch nasadenia modelov.

1.3 Ciele a úlohy práce

Hlavným cieľom je navrhnúť a experimentálne overiť parameterovo efektívnu metódu adaptácie veľkých jazykových modelov na nový jazyk (slovenský) prostredníctvom algoritmu XGBLoRA. Tento cieľ sa delí na tieto čiastkové úlohy:

1. Preskúmať algoritmus XGBLoRA a príbuzné PEFT-metódy (LoRA, AdaLoRA, FanLoRA, LoRA-XS, OLoRA); analyzovať ich silné a slabé stránky a odôvodniť výber XGBLoRA pre riešený problém.

2. Implementovať XGBLoRA pre zvolený model a integrovať ho do existujúceho frameworku; umožniť flexibilné nastavenie parametrov (počet iterácií boostingu, rank, frekvencia zlúčenia adaptérov atď.).

3. Pripraviť experimenty doladenia na datových sadách GLUE (angličtina) a SKQuAD (slovenčina); zaznamenať hyper-parametre a priebežné metriky.

4. Prispôsobiť tréning pre slovenský jazyk, vyhodnotiť úspešnosť a prípadne upraviť konfiguráciu (napr. počet epoch, numerickú presnosť) pre stabilnejšie učenie pri menšom objeme dát.

1.4 Štruktúra práce

Bakalárska práca je rozdelená do šiestich kapitol:

- **Kapitola 1 (Úvod)** – motivácia, kontext, ciele a štruktúra práce.

- **Kapitola 2** – prehľad moderných metód adaptácie LLM; opis LoRA a jej variantov (XGBLoRA, AdaLoRA, FanLoRA, LoRA-XS, OLoRA) a ich porovnanie.
- **Kapitola 3** – implementácia XGBLoRA: architektúra knižnice, kľúčové triedy a mechanizmy iteratívneho učenia adaptérov.
- **Kapitola 4** – venovaná problémom jazykovej adaptácie modelov, najmä pomocou LoRA-podobných metód; obsahuje teoretickú analýzu a opis realizovaných experimentov.
- **Kapitola 5** – hodnotenie výsledkov experimentov; porovnanie plného doladenia, LoRA a XGBLoRA na GLUE a SKQuAD, diskusia typov chýb a možných príčin.
- **Kapitola 6** – záver; zhrnutie prínosov XGBLoRA, možnosti praktického nasadenia a návrhy ďalšieho výskumu (napr. spojenie s kvantizáciou, väčšie modely, iné jazyky či domény).

Prácu uzatvára zoznam použitej literatúry a prílohy s dodatočnými výsledkami a ukážkami implementácie.

Kapitola 2

Moderné metódy adaptácie LLM

V tejto kapitole bude vykonaná analýza moderných metód adaptácie LLM založených na LoRA. Budú medzi sebou porovnané a zároveň budú navrhnuté možné zlepšenia.

2.1 Zásady nízkorankovej adaptácie (LoRA)

2.1.1 Úvod

Adaptácia veľkých jazykových modelov (Large Language Models, LLM) na konkrétne úlohy často naráža na problém obrovského počtu parametrov, čo si vyžaduje značné výpočtové zdroje a pamäť, najmä pri použití pokročilých optimalizátorov ako Adam. Úplné dolaďovanie všetkých parametrov je často nemožné alebo neefektívne, najmä ak je potrebné prispôsobiť model na množstvo rôznych úloh alebo jazykov. Na riešenie tohto problému sa rozvíjajú metódy parameter-efektívneho dolaďovania (Parameter-Efficient Fine-Tuning, PEFT), ktoré umožňujú nastaviť model pri zmene len malej časti jeho parametrov. Existuje niekoľko tried PEFT-metód:

1. Pridanie malých adaptačných modulov do vrstiev modelu (napríklad adaptéry podľa Houlsby a kol., alebo nízko-rankové prídavky LoRA),

2. Vkladanie trénovateľných tokenov do vstupov (prompt tuning, prefix tuning),

3. Dolaďovanie iba vybraných existujúcich parametrov, atď.

V tejto práci sa zameriavame najmä na metódu LoRA (Low-Rank Adaptation) a viaceré jej moderné modifikácie, ktoré sa osvedčili ako účinné riešenia na dolaďovanie LLM s minimálnymi zdrojmi. Najviac priestoru venujeme modifikácii XGBLoRA. V tejto kapitole prinášame analýzu týchto LoRA-modifikácií a ich charakteristiky.

2.1.2 LoRA

LoRA ponúka reprezentovať zmenu váh neurónovej siete ako súčin matíc s nízkou hodnotou ranku. Formálne, ak je $W \in \mathbb{R}^{d\times k}$ trénovaná váhová matica, namiesto jej úplnej aktualizácie trénujeme dve oveľa menšie matice $A \in \mathbb{R}^{d\times r}$ a $B \in \mathbb{R}^{r\times k}$ (kde $r \ll \min(d,k)$), takže adaptácia $\Delta W = AB$ sa pripočíta k pôvodným (zmrazeným) váham:

$$W_{\text{new}} = W_{\text{pretrained}} + \Delta W.$$

Počet trénovaných parametrov je tak približne $(d+k)\times r$, čo predstavuje iba zlomok celkových parametrov modelu (zvyčajne $< 0{,}1\,\%$ pre $r \sim 8$).

Struktúra adaptácie LoRA: veľká váhová matica (znázornená ružovou farbou) ostáva zmrazená, zatiaľ čo malá nízko-ranková podsieť (znázornené oranžovými maticami) sa trénuje a pridáva k pôvodným váham. *(Obrázok)*

LoRA sa ukázala ako jednoduchý a efektívny spôsob dolaďovania veľkých jazykových modelov, pričom kvalita je porovnateľná s plnou adaptáciou a zároveň sa mení menej ako $1\,\%$ parametrov modelu [1]. Základná verzia LoRA však má určité obmedzenia, napríklad pevne stanovený rank a rovnomerné rozdelenie tohto ranku medzi všetky vrstvy bez ohľadu na rozdielnu dôležitosť jednotlivých vrstiev. Moderné modifikácie LoRA sa snažia odstrániť tieto obmedzenia a zlepšiť kvalitu aj efektivitu.

2.2 Opis metód založených na LoRA

2.2.1 XGBLoRA

Extreme Gradient Boosting LoRA (XGBLoRA) [2] je prístup, ktorý spája LoRA s myšlienkou ansámblov (gradientového boostingu) na postupné trénovanie viacerých nízko-rankových adaptácií. Kým pri štandardnej LoRA s rankom r sa všetkých r bázových ‘zložiek‘ adaptácie učí naraz, v XGBLoRA sa trénuje iteratívna postupnosť nezávislých nízko-rankových adaptácií, pričom každá sa snaží vylepšiť výsledok predchádzajúcich. Tento postup je inšpirovaný gradientovým boostingom, kde množina ‘slabých‘ modelov postupne odstraňuje vzájomné chyby. Viacero adaptácií sa nakoniec zhrnie (sčíta) do silnej celkovej adaptácie s efektívnym rankom $r_{\text{eff}} = T$ (ak prebehlo T iterácií).

Základná idea V štandardnej LoRA (rank r) sa trénuje jedna nízko-ranková matica $\Delta W = AB$. V XGBLoRA dostávame postupnosť adaptácií $\Delta W^{(1)}, \Delta W^{(2)}, \ldots, \Delta W^{(T)}$, kde každá $\Delta W^{(t)} = A^{(t)}B^{(t)}$ má rank 1. Výsledný adaptovaný model využíva súčet všetkých príspevkov:

$$W_{\text{adapted}} = W_{\text{pretrained}} + \sum_{t=1}^{T} \Delta W^{(t)}.$$

Na každom kroku t sa $\Delta W^{(t)}$ učí zohľadniť zostatkovú chybu modelu, berúc do úvahy už vynaložené opravy $\sum_{i<t} \Delta W^{(i)}$.

Algoritmus Pseudokód XGBLoRA môže vyzerať nasledovne:

1. **Inicializácia:** Nastavíme $\Delta W := 0$.
2. **Pre** $t = 1 \dots T$:
   (a) Inicializovať $A^{(t)}, B^{(t)}$ (rank 1) malými náhodnými hodnotami.
   (b) Trénovať $A^{(t)}, B^{(t)}$ na dátach, minimalizujúc chybu, pričom váhy modelu sú $W_0 + \Delta W + A^{(t)}B^{(t)}$. (Parametre W_0 a doterajšie ΔW ostávajú zmrazené.)
   (c) Pripočítať k ΔW: $\Delta W := \Delta W + A^{(t)}B^{(t)}$.

Celkovo tak vzniká adaptácia ranku T, ale trénuje sa iteratívne po jednom rank-1 module. Autori [2] ukazujú, že XGBLoRA stabilne prekonáva štandardnú LoRA pri rovnakom či menšom počte trénovaných parametrov. Metódu možno ďalej využiť na multijazykové či multimodálne nastavenia - pri obmedzenom rozpočte ranku je možné postupne pridávať adaptácie na rôzne jazyky alebo úlohy a zastaviť sa, keď kvalita dosiahne požadovanú úroveň.

2.2.2 AdaLoRA

Adaptive Low-Rank Adaptation (AdaLoRA) [3] je modifikácia LoRA, ktorá navrhuje dynamicky prerozdeliť ‘rozpočet‘ nízkeho ranku medzi rôzne vrstvy modelu v priebehu učenia. V klasickej LoRA je rank r zvyčajne pevný a rovnaký pre všetky matice (napr. $r = 8$ pre W_Q, W_K, W_V a pod.), hoci niektoré vrstvy môžu potrebovať viac parametrov na dobrú adaptáciu a iné zas menej. AdaLoRA rieši tento problém automatickým presúvaním kapacity ranku tam, kde je to najviac potrebné, a orezávaním (nulovaním) menej dôležitých častí adaptácie.

Kľúčové myšlienky

- **Dynamické rozdeľovanie ranku:** Rozpočet ranku je považovaný za obmedzený zdroj, ktorý sa počas tréningu prerozdeľuje v závislosti od dôležitosti jednotlivých komponentov. Niektoré vrstvy môžu dostať väčšiu časť ranku, iné menšiu.
- **SVD parametrizácia:** Inkrementálne aktualizácie ΔW sa vyjadrujú v tvare $P\Lambda Q$, kde P, Q sú (približne) ortonormálne matice a Λ je diagonálna matica singulárnych hodnôt. V priebehu učenia sa menej dôležité singulárne hodnoty orežú (vynulujú).

Vďaka tomu môže model alokovať viac kapacity adaptácie tam, kde je jej prínos väčší, a eliminovať menej dôležité zložky. Empirické výsledky [3] potvrdzujú, že AdaLoRA dokáže pri rovnakom alebo menšom celkovom ranku dosiahnuť kvalitu vyššiu než bežná LoRA.

2.2.3 FanLoRA

FanLoRA [4] sa zameriava na prípad, keď sa v jednom LLM môžu používať viaceré LoRA-moduly paralelne (napr. multi-task scenár alebo multi-tenant, kde každý užívateľ má vlastný LoRA). Viacero LoRA-prídavkov v rôznych vrstvách môže spomaliť inferenciu, pretože pri generovaní textu sa vykonáva viacero maticových násobení. FanLoRA teda navrhuje zredukovať počet LoRA-modulov, pričom si uchová čo najvyššiu kvalitu.

Štvorfázový postup FanLoRA

1. **Plné trénovanie LoRA-modulov:** Najprv sa do všetkých lineárnych vrstiev pridajú LoRA-moduly a natrénujú sa na veľkej dátovej sade.
2. **Vyčíslenie dôležitosti** (tzv. AB-score): Každý modul sa dočasne vyradí a sleduje sa zhoršenie (alebo zlepšenie) výkonu na validačnej sade.
3. **Výber najdôležitejších modulov:** Nechať len tie moduly, ktoré prinášajú najväčší prínos. Zvyšné sa odstránia.
4. **Finálne dolaďovanie:** S vybranými modulmi sa model ďalej trénuje na cieľovej úlohe.

AB-skóre. FanLoRA zavádza metriku AB-skóre pre adaptér m vo vrstve l, označovanú ako $V_{m,l}$. Formálne sú uvažované štyri varianty modelu: (a) kompletný model so všetkými adaptérmi (All), (b) model bez adaptéra (m, l), (c) model len s adaptérom (m, l), (d) model bez adaptéra (m, l), ale so všetkými ostatnými. Autori uvádzajú analógiu s atribučnými metódami, pričom samotné AB-skóre sa v konečnom dôsledku počíta približne ako rozdiel strát medzi modelmi s adaptérom a bez neho, normalizovaný určitým spôsobom. Hodnota $V_{m,l} > 0$ znamená, že adaptér zlepšuje kvalitu, zatiaľ čo $V_{m,l} < 0$ znamená, že adaptér kvalite škodí.

Výsledky a efektivita. FanLoRA umožňuje redukovať počet adaptérov, pričom výsledný model si zachováva alebo dokonca zlepšuje kvalitu v porovnaní s pôvodnou konfiguráciou LoRA obsahujúcou všetky adaptéry, pričom zároveň dochádza k zrýchleniu inferencie. Ak pôvodná LoRA pridávala približne 30% dodatočných výpočtov počas generovania (kvôli mnohým maticovým operáciám A a B v každej vrstve), FanLoRA tento výpočtový overhead výrazne znižuje odstránením nepotrebných modulov.

Táto vlastnosť robí metódu atraktívnou pre praktické aplikácie, kde jedna veľká jazyková modelová inštancia obsluhuje veľké množstvo rôznorodých požiadaviek, pričom je dôležité minimalizovať latenciu.

Uplatniteľnosť. FanLoRA je obzvlášť vhodná v situáciách, keď je súčasne aktívnych viacero adaptérov LoRA (multitasking alebo multiužívateľský scenár na jednom serveri). Pri viacjazyčných modeloch to znamená, že pre každý jazyk sa ukladá osobitný adaptér, pričom pri obsluhe rôznych jazykových požiadaviek dochádza k prepínaniu medzi nimi. FanLoRA by v tomto prípade mohla pomocou spoločného viacjazyčného datasetu identifikovať, ktoré vrstvy či komponenty sú pre adaptáciu jednotlivých jazykov najdôležitejšie, a ponechať napríklad iba jeden adaptér na vrstvu namiesto pôvodných štyroch.

Z experimentov vyplýva [4], že FanLoRA často zachováva alebo dokonca zlepšuje kvalitu oproti pôvodnej (plnej) konfigurácii LoRA a súčasne znižuje čas inferencie.

2.2.4 LoRA-XS

Myšlienka metódy. LoRA-XS (*Low-Rank Adaptation with eXtremely Small number of parameters*) je metóda radikálne znižujúca počet trénovateľných parametrov adaptácie tým, že využíva vopred vypočítané bázy singulárnych vektorov. Na rozdiel od štandardnej metódy LoRA, kde sa trénujú obe faktorizačné matice A a B, pri LoRA-XS sa tieto matice určia vopred a ostanú fixné, zatiaľ čo trénovateľná je len veľmi malá matica rozmeru $r \times r$. Intuícia vychádza z pozorovania, že smery aktualizácie váh navrhované metódou LoRA často korelujú s už existujúcimi hlavnými komponentami samotných váh pôvodného modelu. Inak povedané, LoRA spravidla „zvýrazňuje“ tie smery v priestore váh, ktoré sú už prítomné v predtrénovanom modeli. Preto je rozumné tieto hlavné smery (získané pomocou SVD pôvodnej váhovej matice) použiť ako bázu pre adaptáciu, pričom model môže škálovať a kombinovať len tieto preddefinované smery.

Pre každú váhovú maticu W, ktorú chceme adaptovať, sa postupuje nasledovne:

1. Vykoná sa klasický rozklad SVD:

$$W \approx U\Sigma V^T, \quad U \in \mathbb{R}^{d\times r},\ V^T \in \mathbb{R}^{r\times k},$$

   kde U a V^T obsahujú r najväčších singulárnych komponentov váhy a Σ zodpovedajúce singulárne hodnoty.

2. Matice U a V^T sú fixované (neaktualizujú sa počas trénovania). Hrajú úlohu matíc A a B z LoRA, no nemenia sa, pretože už reprezentujú hlavné smery pôvodných váh.

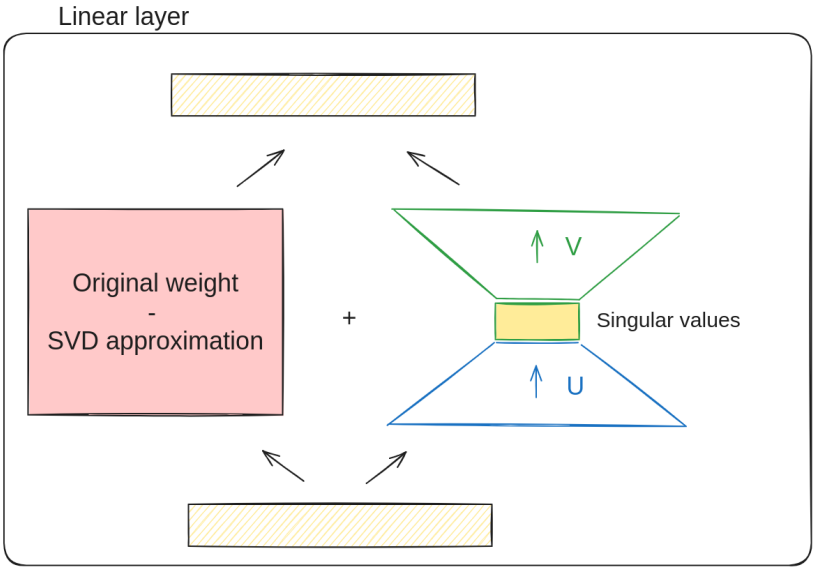
3. Zavádza sa malá trénovateľná matica $R \in \mathbb{R}^{r \times r}$, ktorá sa umiestňuje medzi U a V^T. Adaptácia je potom definovaná ako:

$$\Delta W = URV^T$$

Keďže r je malé číslo, matica R obsahuje iba r^2 parametrov – čo je zanedbateľne málo v porovnaní s pôvodným počtom parametrov $d \times k$ alebo aj v porovnaní s $(d + k)r$ v klasickej LoRA. Napríklad pre $d = k = 1024$ a $r = 8$ má LoRA približne 16 tisíc parametrov na vrstvu, zatiaľ čo LoRA-XS iba $8 \times 8 = 64$.

4. Matica R sa obvykle inicializuje nulovou alebo takmer nulovou maticou (prípadne jednotkovou či Σ). Tým je zaistené, že na začiatku $\Delta W = U0V^T = 0$ a teda model sa nezmení.

5. Počas trénovania sa aktualizuje iba matica R, zatiaľ čo pôvodné váhy W a matice U, V ostávajú nemenné.



Obr. 2.1: Schéma SVD-adaptácie (analóg LoRA-XS)
: pôvodná váha (ružová) je rozložená na fixované matice U (modrá), V (zelená) a trénovateľné „singulárne hodnoty“ (matica R, žltá). LoRA-XS priamo používa takéto rozloženie: $\Delta W = URV^T$, pričom matice U, V sú získané pomocou SVD a netrénujú sa, a jediným trénovateľným komponentom je matica R.

V pôvodnej práci bolo ukázané, že LoRA-XS môže redukovať počet trénovateľných parametrov rádovo o desiatky až stovky krát oproti klasickej LoRA, pričom kvalita na mnohých úlohách (GLUE, MATH a ďalšie) sa neznižuje, a v niektorých prípadoch dokonca zvyšuje. Znamená to, že hlavné singulárne vektory skutočne zachytávajú veľkú

časť významného priestoru adaptácie a malá matica R dokáže tieto smery dostatočne efektívne kombinovať pre prispôsobenie modelu novej úlohe.

LoRA-XS má však menšiu flexibilitu: ak žiadúca aktualizácia ΔW obsahuje komponenty mimo zvoleného podpriestoru, model ich nebude schopný vyjadriť. Preto je výber vhodného ranku r kľúčový – musí byť dostatočne veľký na zachytenie relevantných charakteristík novej úlohy. V opačnom prípade môže dôjsť k nedostatočnému prispôsobeniu. Autori uvádzajú, že pri dostatočne zvolenom ranku r je metóda stabilne porovnateľná s klasickou LoRA a ďalšími novšími metódami (napríklad VeRA).

Treba tiež spomenúť príbuznosť myšlienok metód AdaLoRA a LoRA-XS, pretože obe využívajú SVD reprezentáciu váh. Rozdiel spočíva v tom, že AdaLoRA stále trénuje matice P a Q (analóg U, V) a dynamicky upravuje ich rank počas trénovania, zatiaľ čo LoRA-XS fixuje matice U, V a minimalizuje počet trénovaných parametrov na malú maticu R. Teoreticky je možné kombinovať obe metódy, napríklad určením vysokorankových matíc U, V pomocou SVD a následným adaptívnym prerezávaním R-matice podľa princípov AdaLoRA.

2.2.5 OLoRA

Základná myšlienka. Metóda OLoRA (*Orthonormal LoRA*) nemení samotnú architektúru adaptácie, ale zameriava sa na zlepšenie procesu učenia pomocou špeciálnej inicializácie váhových matíc. V štandardnej LoRA sa matice A a B často inicializujú náhodne – napríklad A náhodnou maticou s malou varianciou a B ako nulová matica, aby bolo zaistené, že $\Delta W = 0$ na začiatku trénovania. Aj keď takáto inicializácia neovplyvní kvalitu modelu na štarte, nemusí dostatočne efektívne využívať dostupný parameterický priestor – napríklad smery v A môžu byť silno korelované a hodnoty v B príliš malé.

OLoRA navrhuje inicializovať matice A a B ako ortonormálne, čo im poskytne lepšie mierky, nezávislosť a urýchli konvergenciu trénovania. Okrem toho to môže pozitívne ovplyvniť výslednú kvalitu modelu.

Technické riešenie. Na inicializáciu matíc A a B využíva OLoRA QR-rozklad. Napríklad, pre $A \in \mathbb{R}^{d\times r}$ sa najprv vygeneruje náhodná matica A_0, na ktorú sa aplikuje QR-dekompozícia. Prvých r ortonormovaných stĺpcov Q sa použije ako počiatočná hodnota A. Analogicky sa ortonormalizuje aj $B^T \in \mathbb{R}^{r\times k}$, pričom výsledkom je, že $A^T A \approx I_r$ a $BB^T \approx I_r$ už na začiatku trénovania.

Takáto inicializácia zabezpečuje, že zmeny váh počas trénovania budú prebiehať v vzájomne nezávislých smeroch, čo zlepšuje efektivitu gradientného zostupu. Okrem toho, ortonormalita pomáha zachovať spektrálne vlastnosti aktualizácie ΔW – žiadny smer v priestore nie je dominantný alebo „vypnutý“. Všetkých r komponentov LoRA sú teda na začiatku rovnocenné a pripravené na učenie.

Efektivita a využitie. OLoRA v skutočnosti nezavádza žiadne nové parametre a nemení počet výpočtov pri inferencii – ide čisto o metódu inicializácie. Na rozdiel od ostatných modifikácií LoRA preto neovplyvňuje parameterickú efektivitu, ale výrazne zlepšuje efektivitu učenia. Možno ho kombinovať s ľubovoľným variantom LoRA – napríklad s adaptívnymi (AdaLoRA) alebo boostingovými (XGBLoRA) metódami. Napríklad v AdaLoRA sa tiež využíva myšlienka ortogonalizácie matíc P a Q, aj keď prostredníctvom regularizačného člena a nie explicitnej inicializácie.

2.2.6 Porovnanie metód

Nižšie je tabuľka, ktorá sumarizuje kľúčové vlastnosti uvedených metód v porovnaní so štandardnou LoRA:
Z tabuľky vidno, že jednotlivé prístupy sa zameriavajú na rôzne aspekty efektivity:

- **XGBLoRA** zlepšuje kvalitu pri rovnakom počte parametrov, a to vďaka postupnému osvojeniu ranku (efekt ansámblu).
- **AdaLoRA** optimálne využíva pevný rozpočet ranku a automaticky hľadá dôležité miesta adaptácie.
- **FanLoRA** znižuje náklady na inferenciu pri viacerých adaptéroch naraz, čo je užitočné v priemyselných aplikáciách.
- **LoRA-XS** extrémne znižuje pamäť na adaptéry a umožňuje ukladať či nasadzovať veľké množstvo individuálnych adaptácií.
- **OLoRA** zrýchľuje a zjednodušuje tréning (lepšia inicializácia), čo môže nepriamo umožniť využiť väčší rank alebo väčší dataset.

2.2.7 Zhrnutie

Moderné modifikácie LoRA podstatne rozširujú možnosti parameter-efektívneho dolaďovania veľkých jazykových modelov. V kontexte multijazykových úloh môžu tieto prístupy priniesť osobitné výhody. Napríklad:

- **AdaLoRA** dokáže prerozdeliť parametre adaptácie medzi rôzne jazyky podľa ich náročnosti,
- **FanLoRA** minimalizuje spomalenie inferencie pri podpore viacerých jazykov naraz,
- **LoRA-XS** umožňuje, aby boli adaptácie na každý jazyk dostatočne malé a zmestili sa napr. priamo do prehliadača či mobilu,

Metóda	**Počet trénovaných parametrov (voči LoRA)**	**Rank a jeho použitie**	**Tréning / inicializácia**	**Inferencia (oneskorenie)**
LoRA	Základná hodnota (100%)	Fixný r pre všetky vrstvy	Bežné učenie A, B; náhodná inici.	Mierne zvýšenie času (doplnkové maticové násobenia)
XGBLoRA	$\sim$ ako LoRA s ekv. rankom T (spolu 100%)	Efektívny rank $= T$ (súčet rank-1)	Iteratívne učenie T adaptérov ranku 1	Nezvyšuje oneskorenie (možno zlúčiť do jedného adaptéra)
AdaLoRA	Môže byť nižší bez straty kvality	Rank dynamický, prerozdelený medzi vrstvy	Špeciálny algoritmus s SVD; periodické orezávanie; ortogon. regularizácia	Ako LoRA (rovnaký počet operácií pri výstupe)
FanLoRA	Porovnateľné s LoRA (po odfiltrovaní)	Rank každého modulu fixný, ale menej modulov	Dvojfázové učenie: najprv plná adaptácia, potom výber a ďalší tréning	Výrazne nižšie oneskorenie (menej aktívnych adaptérov)
LoRA-XS	Výrazne znížený (až $< 1\%$ voči LoRA)	Fixný, malý r; rámec U, V	Vyžaduje SVD pôvodných váh; trénuje sa len R $(r \times r)$	Ako LoRA (pridáva iba malé násobenie R)
OLoRA	Rovnako ako LoRA	Fixný r	Počiatočná ortonormálna inici. (QR); rýchlejšie učenie	Ako LoRA (bez vplyvu na rýchlosť výstupu)

Tabuľka 2.1: Porovnanie vybraných modifikácií LoRA podľa hlavných kritérií.

- **XGBLoRA** dosahuje vysokú kvalitu postupným rozširovaním adaptácie, ak je náročné hneď vyhradiť veľký rank,
- **OLoRA** môže urýchliť učenie v kombinácii s ďalšími technikami (napr. AdaLoRA).

Tieto metódy tak ponúkajú komplexný súbor nástrojov na vytváranie výkonných, prispôsobených a pamäťovo efektívnych riešení pre veľké jazykové modely v rôznych kontextoch.

Kapitola 3

Implementácia XGBLoRA

V tejto kapitole bude detailne opísaný proces implementácie metódy XGBLoRA.

3.1 Architektúra knižnice a štruktúra kódu

Repozitár je organizovaný ako modul `gm`. Vo vnútri `gm` je kód rozdelený na podmoduly podľa funkcionality:

3.1.1 `gmlayers`

Definuje vrstvy (moduly) potrebné pre fungovanie knižnice.

- `pseudo_layers` - obsahuje 'pseudo-vrstvy' (vrstvy, ktoré vo svojej triede neobsahujú váhy) a úložisko váh, ktoré zároveň plní úlohu vrstvy uchovávajúcej všetky váhy pre zvolené pseudo-vrstvy. Konkrétne sú tu definované:
  - `PseudoLayer` – základná trieda pre všetky pseudo-vrstvy; dedia po nej triedy, ktoré definujú vlastnosť `.shapes` – zoznam rozmerov tenzorov danej vrstvy.
  - `PseudoLinear` – podtrieda `PseudoLayer` určená na nahradenie bežnej lineárnej vrstvy (`nn.Linear`).
  - `PseudoConv1D` – analóg pre 1D konvolúciu (framework HuggingFace používa `Conv1D` ako lineárnu vrstvu v niektorých modeloch, napr. GPT-2).
  - ďalšie pseudo-vrstvy, napríklad `PseudoEmbedding`, `PseudoLayerNorm`.

3.1.2 `weights_storage`

Podsystém na ukladanie (a aktualizáciu) váh – slúži ako vrstva-úložisko všetkých váh pre vybrané pseudo-vrstvy:

- `WeightsStorage` – základná trieda úložiska váh, ktorá spravuje zoznam vrstiev a ich parametrov. Metóda `forward` vracia váhy pre zvolenú pseudo-vrstvu a umožňuje aplikovať ľubovoľnú predspracovateľskú logiku ešte pred ich použitím vo výpočtoch.
- `LoRAWeightsStorage` – podtrieda `WeightsStorage` rozšírená o podporu LoRA adaptérov. Vďaka možnosti manipulovať s váhami pred ich použitím možno k nim pripočítať LoRA korekcie. Trieda obsahuje dátovú štruktúru pre LoRA (zoznam LoRA modulov pre každú vrstvu) a implementuje sčítanie hlavných váh s LoRA -váhami počas `forward` priechodu. Navyše poskytuje metódy `apply_weights`, `reset_lora`, `enable_grad`/`disable_grad` na správu adaptérov.
- `LoraWeightsStorageConfig` – konfiguračná trieda odovzdávaná do `LoRAWeightsStorage`; určuje stratégie inicializácie/aktivácie, zariadenie pre uloženie váh a i. Počas vývoja rástol počet argumentov odvodených tried od `WeightsStorage`, preto bolo rozhodnuté vytvoriť osobitnú triedu opisujúcu konfiguráciu úložiska váh.

3.1.3 `gmlora`

Implementácia samotnej logiky LoRA/XGBLoRA (okrem pripočítania korekcií k hlavným parametrom modelu – to je implementované v `LoRAWeightsStorage`):

- triedy LoRA adaptérov, napríklad `LoRA` a základná `BaseLoRA`;
- stratégie aktivácie adaptérov (*enable strategies*), napríklad `LoraEnableAllStrategy` pre simultánne zapnutie všetkých adaptérov a `WeightedSubsetEnableStrategy` pre zapnutie podmnožiny ako v XGBLoRA;
- stratégie inicializácie LoRA (*init strategies*), napríklad `LoRAFullInitStrategy` pre umiestnenie adaptérov do všetkých vrstiev;
- ďalšie rôzne stratégie aktivácie a inicializácie, napríklad `RandomSubsetEnableStrategy`, `GuaranteedUnderusedSubsetEnableStrategy` – alternatívne prístupy výberu adaptérov, a zastaraná trieda `XgbLoraEnableStrategy`.

3.1.4 `gm/pseudo_model.py`

Obsahuje triedu `PseudoModule` a logiku „patchovania“ (nahrádzania) vrstiev modelu pseudo-vrstvami s LoRA:

- `PseudoModule` dedí po `nn.Module` a uchováva pôvodný modul (model) a objekt `WeightsStorage`.

- Statická metóda `PseudoModule.create_patched_pseudo_model` prechádza rekurzívne vrstvami pôvodného modelu a nahrádza určené vrstvy ekvivalentnými pseudo-vrstvami:
  - parameter `mapping` definuje korešpondenciu pôvodnej triedy vrstvy a pseudo-vrstvy (napr. `nn.Linear`: `PseudoLinear`, `Conv1D`: `PseudoConv1D`);
  - parameter `target_modules` umožňuje filtrovať podľa názvu len tie vrstvy, na ktoré sa má aplikovať LoRA. V testoch napríklad `target_modules=,'query','key'` znamená, že spomedzi všetkých `nn.Linear` sa na `PseudoLinear` nahradia iba tie, ktorých názov obsahuje podreťazec „query“, „key“ alebo „value“. Takýto filter vyberie lineárne podvrstvy samo-pozornosti (projekcie Q, K, V) a ostatné lineárne vrstvy (napr. vo výstupnom klasifikátore alebo v medzivrstve FFN) ponechá nedotknuté. Ide o štandardný prístup LoRA – pripájať adaptéry iba na vybrané váhy (zvyčajne matice pozornosti).
  - pri nahrádzaní sa pre každý vhodný podvrstvu volá `PseudoLinear.from_module`, ktorý vytvorí príslušnú pseudo-vrstvu a zaregistruje ju v úložisku váh. Pôvodné parametre vrstvy sa odovzdávajú do `weights_storage.set_parameters` na inicializáciu úložiska týmito hodnotami.

3.2 Mechanizmus `PseudoModule` a `PseudoLayer` pre integráciu LoRA

Náhrada vrstiev za pseudo-vrstvy. Základná myšlienka integrácie LoRA v knižnici `gm` je nemodifikovať priamo triedy modelov HuggingFace, ale nahradiť ich za moduly kompatibilné s rozhraním. Funkcia `PseudoModule.create_patched_pseudo_model` rekurzívne prechádza vnorené `nn.Module` pôvodného modelu a vykonáva ich výmenu.

Napríklad v `test_roberta_xgblora.py` po načítaní predtrénovaného modelu:

```
pseudo_model = PseudoModule.create_patched_pseudo_model(
weights_storage=weights_storage,
module=model,
mapping={nn.Linear: PseudoLinear, Conv1D: PseudoConv1D},
target_modules=['query', 'key', 'value'],
)
```

Tu `weights_storage` je vopred vytvorená inštancia `LoRAWeightsStorage`. Funkcia prejde celý model a pre každý podmodul:

- Ak je trieda podmodulu `nn.Linear` a jeho názov obsahuje `'query'`, `'key'` alebo `'value'`, podmodul bude nahradený: namiesto neho sa do objektu modelu vloží objekt `PseudoLinear`.

- V opačnom prípade sa rekurzívne spracujú jeho dcérske moduly (táto rekurzia klesá hierarchiou modelu).

3.2.1 Registrácia váh v úložisku

Keď sa vytvorí nový `PseudoLinear` cez `PseudoLinear.from_module`, vykoná nasledovné:

- Vypočíta pseudo-tvary - rozmery tenzorov, ktoré by mala táto vrstva uložiť. Pre lineárnu vrstvu s váhovým parametrom W veľkosti `[out_features, in_features]` a prípadne posunom b veľkosti `[out_features]`, vytvorí zoznam

  `shapes = [torch.Size([out_feat, in_feat]), torch.Size([out_feat])]`

  alebo bez druhého, ak bias chýba.

- Zavolá `self.register_layer()`, zdedenú metódu z `PseudoLayer`. Táto metóda zaregistruje vrstvu v `WeightsStorage`:
  - Zavolá `weights_storage.add_layer(self)` a uloží vrátený index do `self._storage_index`.
  - `WeightsStorage.add_layer` následne pridá objekt vrstvy do zoznamu `self._layers` a vytvorí *placeholder* pre parametre požadovaných rozmerov na meta-zariadení.
  - *Placeholder* je `nn.Parameter` so správnym tvarom, ale umiestnený na `torch.device("meta")` (zaberajúci nulovú pamäť), čo umožňuje rezervovať miesto pre skutočné váhy, ktoré sa inicializujú neskôr.

- Potom `PseudoLayer.from_module` zavolá `weights_storage.set_parameters(layer_index, list(module.parameters()))`. To znamená, že vezme pôvodné parametre originálnej vrstvy (predtrénovanú maticu W a b) a skopíruje ich do úložiska namiesto placeholderov. Vo vnútri `WeightsStorage.set_parameters` sa vykoná kontrola zhody rozmerov a výmena placeholder-parametrov za skutočné objekty `param`, pričom sa zároveň zaregistrujú ako parametre modulu `WeightsStorage` (aby ich optimalizátor videl, ak budú trénovateľné). V prípade LoRA budú základné váhy hneď po tomto kroku takmer všetky zmrazené (gradienty vypnuté), ale počas fázy patchovania prebieha iba ich presun do `WeightsStorage`.

3.2.2 Forward cez pseudo-vrstvu

Pseudo-vrstva musí fungovať presne ako originálna (`nn.Linear`), ale váhy nesmie brať zo svojho atribútu, ale z úložiska, pričom zohľadňuje prídavok LoRA. Toto je vyriešené

prepísaním metódy `__call__` v `PseudoLayer` a použitím `WeightsStorage.__call__` (alebo metódy `forward`).

Mechanizmus funguje nasledovne:

- Každý `PseudoLayer` uchováva odkaz `self._weights_storage` a svoj `storage_index`.
- Objekt `WeightsStorage` má implementovanú metódu `forward(layer_index)`, ktorá vráti zoznam aktuálnych parametrov pre danú vrstvu.
- V základnom `WeightsStorage` to jednoducho znamená vrátiť uložené parametre (v našom prípade zoznam `[W, b]` pre danú vrstvu).
- V `LoRAWeightsStorage` je metóda `forward` prepísaná: sčíta uloženú základnú váhu s delta-váhou z LoRA, ak je príslušný adaptér zapnutý.
- V podstate teda vracia `[W + \Delta W, b]` namiesto len `[W, b]` (pričom sa predpokladá, že na bias sa LoRA neuplatňuje, takže bias sa vráti bez zmeny).
- V `PseudoLayer.__call__` (v rodičovskej triede pre všetky `PseudoLinear` a podobné) je to urobené tak, že pri zavolaní pseudo-vrstvy si najprv vytiahne váhy z úložiska a až potom zavolá štandardný `forward` svojej triedy:

```
def __call__(self, *args, **kwargs):
# uistiť sa, že je vrstva registrovaná
if self._storage_index is None:
raise "call␣register_layer()␣first"
# získať argumenty (prípadne parsovať kwargs, ak treba)
args, kwargs = self._weights_storage.get_argument_parsing_strategy.parse(*
  ↪ args, **kwargs)
# dosadiť váhy z úložiska a zavolať forward
return super().__call__(
self._weights_storage(self._storage_index, **kwargs),
*args,
**kwargs,
)
```

3.3 Trieda adaptéra LoRA a integrácia delta-váh

Teraz sa pozrime na samotnú triedu LoRA adaptéra - čo presne predstavuje objekt, ktorý sa pridáva k váham. V kóde je definovaná základná trieda `BaseLoRA` (v súbore `gm/lora/base_lora.py`) a konkrétna implementácia `LoRA` (v súbore `gm/lora/lora.py`).

Základná myšlienka LoRA je nasledovná: Pre danú maticu váh W_0 veľkosti $d_{out} \times d_{in}$ zavedieme dve malé parametrové matice: A veľkosti $d_{out} \times r$ a B veľkosti $r \times d_{in}$ (kde $r \ll \min(d_{out}, d_{in})$ – rank adaptácie). Počas adaptácie (fine-tuningu) sa predpokladá, že výsledná váha je $W = W_0 + \Delta W$, kde $\Delta W = A \cdot B$. (V pôvodnej práci o LoRA je B down-projekcia a A up-projekcia, ale podstatný je ich súčin.) Parametre W_0 zostávajú zamrazené, zatiaľ čo A a B sa učia, čo výrazne znižuje počet učených parametrov. Zavádza sa tiež škálovací koeficient α, ktorý sa aplikuje na ΔW ako $\Delta W \cdot \frac{\alpha}{r}$, aby sa kontroloval počiatočný vplyv (keď sú A a B inicializované na malé hodnoty). V knižnici `gm` sa hodnota `alpha` predáva spolu s argumentmi pri vytváraní objektu triedy `LoRA`.

V implementácii:

- Trieda `LoRA` (potomok `BaseLoRA`) v konštruktore prijíma `shape` pôvodnej váhy, `rank`, `alpha`, `dtype` a `lora_dropout`.
- Vytvorí potrebné nízkorozmerné matice, ktorých súčin bude mať rozmery zodpovedajúce `shape`.
- Ak `shape` má práve dva rozmery, matice sa inicializujú podľa pôvodnej práce o LoRA[1]: prvá matica (up-projekcia) sa inicializuje náhodne so štandardnou odchýlkou $1/rank$, druhá matica (down-projekcia) sa inicializuje nulami (prípustné sú aj hodnoty blízke nule). V prípade viacrozmerného `shape` (počet rozmerov nie je dva) sa všetky matice inicializujú náhodne. (V tejto práci sa viacrozmerné prípady nevyužívali, ale bola pridaná minimálna podpora všeobecných tvarov.)
- Metóda `compute_lora_delta()` adaptéra počíta $\Delta W = B \times A$ (s ohľadom na α). Je potrebné poznamenať, že v kóde sa nepoužíva explicitné rozdelenie na matice A a B - miesto toho môže adaptér vytvoriť ľubovoľný počet matíc (tenzorov), uložených v poli, aby bolo možné pripojiť adaptér aj na tenzory s tromi a viac rozmermi. (V praxi sa však počas práce takéto scenáre nevyskytli.)
- Okrem toho má adaptér metódy `enable()`, `disable()`, `enable_grad()` a `disable_grad()`, ktoré riadia, či sa adaptér zúčastňuje na výpočtoch a tréningu:
  - `enable()` nastaví interný príznak `self.enabled = True` (inak by ho `LoRAWeightsStorage.forward` ignoroval).
  - `enable_grad()` zapne `requires_grad=True` pre parametre A a B, aby sa mohli trénovať. (Poznámka: pri vypnutom adaptéri je vhodné deaktivovať aj gradienty, aby sa šetrili zdroje.)
  - `reset_matrices()` vynuluje alebo reinicializuje A a B (používa sa napríklad pri reštarte adaptéra v `XGBLoRA`).

[1]LINK=inicializácia LoRA matíc

3.4 Inicializácia úložiska v LoRAWeightsStorage

V momente, keď sú všetky vrstvy zaregistrované (po patchovaní modelu), sa volá metóda `LoRAWeightsStorage.build_storage()`. Pri volaní tejto metódy sa vykoná:

1. **Distribúcia adaptérov vo vrstvách:** Rozdelenie prebieha podľa stratégie typu `BaseLoRAInitStrategy`. V prípade `LoRAFullInitStrategy` sa adaptéry vkladajú pre každú maticu/tenzor každého vhodného modulu, okrem tých, ktoré majú rozmernosť 1 (napríklad bias-vrstvy).

2. **Inicializácia stratégie zapnutia:** Stratégia zapnutia určuje, ktoré LoRA-adaptéry budú v danom okamihu aktívne. Pri klasickej LoRA jednoducho zapíname všetky adaptéry, zatiaľ čo pri `XGBLoRA` sa v danom momente používa iba náhodná tretina adaptérov. Stratégia sa môže inicializovať až po tom, čo je známy presný počet adaptérov (keďže niektoré implementácie stratégie to vyžadujú). Preto bolo rozhodnuté inicializovať ju v rámci `build_storage()`.

3. **Registrácia adaptérov ako submodulov:**

```
for layer_idx, layer_modules in enumerate(self.lora_modules):
for module_idx, module in enumerate(layer_modules):  self.add_module(f"
  ↪ lora{layer_idx}_{module_idx}", module)
```

   Táto registrácia zabezpečuje, že adaptéry sa stanú súčasťou `LoRAWeightsStorage` ako `nn.Module` a ich parametre budú súčasťou `LoRAWeightsStorage.parameters()`, aby ich optimalizátor mohol sledovať (ak `requires_grad=True`).

4. **Uloženie pôvodných parametrov do `LoRAWeightsStorage`:** Volanie `super().build_storage()` uloží pôvodné parametre vrstiev v triede `LoRAWeightsStorage`, keďže pseudo-vrstvy nemajú vlastné váhy a spoliehajú sa na `WeightsStorage`.

5. **Počiatočná konfigurácia LoRA:** Nakoniec sa volá `self.reset_lora()`, ktorý nastaví počiatočný stav adaptérov:

   - Vygeneruje novú aktivačnú masku.
   - Zapne adaptéry, ktoré sú podľa masky aktívne (pomocou `enable()`) a vypne ostatné (pomocou `disable()`).
   - Pre každý aktívny adaptér reinicializuje nízkorozmerné matice.
   - Následne budú iba aktívne adaptéry ovplyvňovať výpočty.

3.4.1 Stav systému na začiatku trénovania

Na začiatku učenia máme:

- V `LoRAWeightsStorage._storage` sú uložené hlavné váhy modelu (zamrazené).
- V `LoRAWeightsStorage._lora_modules` sú uložené zoznamy adaptérov, pričom:
  - V prípade `XGBLoRA` je aktívna iba časť (`enabled=True`),
  - V prípade klasickej LoRA sú aktívne všetky adaptéry.
- Forward hociktorej vrstvy bude pripočítavať delta váhu iba tam, kde je adaptér `enabled=True` (pretože `LoRAWeightsStorage.forward` kontroluje príznak `module.enabled`).
- Všetky aktívne adaptéry majú `requires_grad=True`, zatiaľ čo neaktívne sú deaktivované (`disabled`), takže neovplyvňujú výpočty a nie je nutné sledovať ich gradienty.

3.5 Riadenie adaptérov

3.5.1 Bežný režim (`LoRAEnableAllStrategy`)

Pri fine-tuningu klasickej LoRA sú všetky adaptéry pripojené k modelu aktívne a trénované paralelne. Stratégia `LoRAEnableAllStrategy` jednoducho vracia masku obsahujúcu samé `True`. Jej implementácia je veľmi jednoduchá:

```
class LoraEnableAllStrategy(BaseLoraEnableStrategy):
def __init__(self, adapters_count: int, **kwargs):
super().init(adapters_count)

def get_activation_mask(self) -> List[bool]:
return [True] * self._adapters_count
```

3.5.2 Režim XGBLoRA (`WeightedSubsetEnableStrategy`)

`WeightedSubsetEnableStrategy` je jedna z kľúčových tried pre `XGBLoRA`. Dedí od `BaseLoraEnableStrategy` a implementuje pravdepodobnostný výber podmnožiny indexov adaptérov tak, aby menej používané adaptéry mali väčšiu šancu byť vybrané.

3.5.3 Prepínanie aktívnych adaptérov počas trénovania

Kľúčová vlastnosť `XGBLoRA` je postupné aktivovanie všetkých adaptérov a sčítavanie ich efektov, podobne ako pri gradientnom boostingu, kde nové stromy postupne kompenzujú chyby predchádzajúcich. V našej knižnici sa toto správanie dosahuje volaním metód `apply_weights()` a `reset_lora()` na objekte `LoRAWeightsStorage` počas trénovania.

- `LoRAWeightsStorage.apply_weights()` prechádza všetky aktuálne aktívne adaptéry a aplikuje ich korekcie na základné váhy.

```
for each layer_idx, for each lora_module in self._lora_modules[layer_idx]:
if lora_module is None or not lora_module.enabled:
continue
with torch.no_grad():
self._storage[layer_idx][param_idx] += lora_module.compute_lora_delta()
```

- Tu `self._storage[layer_idx][param_idx]` predstavuje samotný `nn.Parameter` základnej váhy vrstvy (napríklad maticu W), zatiaľ čo `lora_module.compute_lora_delta()` je vypočítaná delta pre túto váhu. Sčítanie sa vykonáva v bloku `torch.no_grad()`, pretože manuálne upravujeme zamrznuté váhy bez potreby sledovania gradientov.

- `LoRAWeightsStorage.reset_lora()` po aplikovaní volá `get_activation_mask()` podľa zvolenej stratégie a pripraví adaptéry na nový cyklus:
  - Pre aktívne adaptéry podľa novej masky znovu inicializuje nízkorozmerné matice pomocou `reset_matrices()`.
  - Aktivuje ich volaním `enable()` a nastaví im `requires_grad=True` volaním `enable_grad()`.
  - Ostatné adaptéry deaktivuje volaním `disable()`.

3.6 Riadenie váh a gradientov počas trénovania

Pokiaľ ide o spôsob riadenia váh a gradientov, je možné zhrnúť nasledovné:

- **Zmrazenie základných váh:** Tak ako pri štandardnej LoRA, aj tu základný model (predtrénované váhy) nesmie meniť svoje hodnoty cez priamy gradient - inak by sa stratil zmysel adaptácie s nízkym rankom. Preto sa po zabalení modelu do `PseudoModule` vypnú gradienty pre všetky pôvodné parametre modelu.

- **Trénovanie s `GradScaler` (AMP):** V testovacích skriptoch sa používa `torch.cuda.amp.GradScaler` pre automatizované zmiešané presné výpočty (AMP). Toto nie je špecifické pre LoRA, ale je to bežná technika na zrýchlenie trénovania na moderných GPU. Dôležité je, že celé naše zasahovanie (LoRA) je kompatibilné s AMP: dodatočné sčítanie `weights + delta` je jednoduchá tenzorová operácia, ktorá takisto funguje v režime half-precision.

- **Akkumulácia gradientov cez adaptéry:** Pred aktualizáciou adaptéru sa ich korekcie pripočítajú k základným váham pôvodného modelu. Tým pádom, aj keď sa každé nové „kolo“ trénovania pracovné matice adaptéru znovu náhodne inicializujú, model postupne kumuluje znalosti nadobudnuté vo všetkých predchádzajúcich kolách.
- **Aktualizácia momentov optimalizátora:** Pri aktualizácii adaptéru je potrebné takisto vynulovať momenty optimalizátora, keďže váhy adaptéru sa menia „manuálne“ - optimalizátor nemá vedomosť o tom, že po aktualizácii (teda po novej inicializácii na začiatku kola) majú adaptéry nové váhy.

Kapitola 4

Adaptácia modelu na slovenské dáta

V tejto kapitole bude najskôr predstavený teoretický prehľad problematiky jazykovej adaptácie veľkých jazykových modelov (LLM) a následne bude opísaný pokus o doladenie LLM na slovenský jazyk pomocou metódy `XGBLoRA`.

4.1 Úvod do jazykovej adaptácie

Adaptácia na nový jazyk je zložitý proces – ako pre človeka, tak aj pre model. Človek si pri učení cudzieho jazyka postupne osvojuje slovnú zásobu a gramatiku, pričom sa opiera o existujúce jazykové skúsenosti. Podobne, veľký jazykový model (LLM), ktorý bol pôvodne trénovaný prevažne na jednom jazyku (zvyčajne angličtine), si musí prispôsobiť svoje váhové parametre špecifikám nového jazyka.

Tento proces však prináša viacero problémov: nedostatok rozsiahlych dátových korpusov v cieľovom jazyku, odlišná morfológia a syntax, nové znaky alebo úplne iný abecedný systém – to všetko sťažuje trénovanie modelu. Napríklad pri jazykoch s bohatou morfológiou (ako hebrejčina alebo arabčina) štandardná tokenizácia navrhnutá pre angličtinu často zlyháva. Bez špeciálnej adaptácie aj výkonné LLM modely dosahujú podstatne horšie výsledky v nízkozásobovaných jazykoch než v angličtine.

Adaptácia LLM na nový jazyk zároveň často naráža na problém tzv. *katastrofického zabúdania* predchádzajúcich znalostí. Kompletné pretrénovanie všetkých parametrov modelu na dátach v novom jazyku môže zhoršiť jeho výkonnosť v pôvodnom jazyku (napr. v angličtine). Zároveň, trénovať všetkých 175+ miliárd parametrov modelu nanovo pre každý jazyk je extrémne nákladné.

Preto sú dnes vysoko aktuálne metódy efektívnej adaptácie, ktoré umožňujú trénovať model len čiastočne, pričom zmeny v pôvodných váhach sú minimálne. V posledných rokoch boli predstavené stratégie tzv. *parameter-efficient fine-tuning* (PEFT) – parameterovo efektívneho dolaďovania, pri ktorých sa netrénuje celá sieť, ale len malé doplnkové moduly alebo špecifické parametre. Medzi takéto prístupy patrí napríklad

Low-Rank Adaptation (LoRA), adaptérové vrstvy, trénovanie promptov a ďalšie techniky. Tieto prístupy umožňujú integrovať nové znalosti do modelu s výrazne nižšími výpočtovými nárokmi.

4.2 Adaptácia LLM na nové jazyky pomocou LoRA a príbuzných metód

Low-Rank Adaptation (LoRA) sa v praxi osvedčila ako jedna z najpopulárnejších PEFT-metód (parameter-efficient fine-tuning) pre adaptáciu veľkých jazykových modelov (LLM). Pri metóde LoRA sa všetky pôvodné váhy modelu zmrazia a do každej vrstvy sa pridajú malé trénovateľné matice nízkeho rádu, ktoré jemne upravujú výpočty. Vďaka tomu je možné efektívne trénovať model na nový jazyk pri učení iba približne ~0.1% parametrov namiesto 100%.

V praxi LoRA výrazne znižuje pamäťové nároky a umožňuje doladiť modely s miliardami parametrov aj na jednej GPU. Napríklad pri adaptácii modelu GPT-3 (175B) metódou LoRA sa počet trénovaných parametrov znížil približne 10 000-krát a potreba GPU pamäte bola trikrát nižšia v porovnaní s plným doladením všetkých parametrov.

LoRA sa úspešne využíva na adaptáciu modelov na nové jazyky. V práci Shmidman et al. (2023) bol predstavený model DictaLM 2.0, ktorý vznikol z anglicko-hebrejského modelu Mistral dlhodobým trénovaním na korpuse približne 200 miliárd tokenov v hebrejčine. Vývojári najskôr rozšírili tokenizátor o približne 1000 nových tokenov pre hebrejčinu a následne doladili model pomocou LoRA adaptérov vo všetkých vrstvách (okrem embedding vrstiev) počas niekoľkých epoch.

Výsledok: výrazné zlepšenie porozumenia a generovania hebrejského textu. Aj po čiastočnom doladení embedding vrstiev a výstupnej hlavy sa kvalita spracovania hebrejčiny citeľne zlepšila. Plné doladenie modelu s LoRA na veľkom hebrejskom korpuse (15 dní tréningu) umožnilo modelu dosiahnuť vynikajúce výsledky v špeciálne vytvorených testoch pre hebrejský jazyk.

Dôležité zistenie tejto štúdie: adaptácia LLM na nový jazyk je možná aj bez kompletného pretrénovania modelu od nuly, a to kombináciou rozšírenia slovníka a metód ako LoRA.

Ďalším príkladom je adaptácia modelov na arabský jazyk. Model Qwen-1.5B od spoločnosti Alibaba bol doladený na arabských dátach pomocou kvantovanej verzie LoRA - QLoRA, čo umožnilo vykonať trénovanie dokonca na GPU s len 4 GB pamäte. Autori skombinovali viaceré arabské datasety (vrátane korpusov Bactrian, OpenAssistant a arabskej Wikipédie) a vyriešili špecifické úlohy - napríklad normalizáciu diakritiky - ešte pred začiatkom trénovania LoRA-adaptérov.

Počas 10 000 krokov trénovania model dosiahol konvergenciu (záverečná strata

~0.1083) a vykazoval zlepšenie vo viacerých úlohách: klasifikácii textu, odpovediach na otázky a určovaní dialektu. To potvrdzuje, že metódy ako `LoRA/QLoRA` otvárajú možnosť adaptácie LLM na jazyky s nízkymi zdrojmi aj v obmedzenom výpočtovom prostredí: *„it's possible to fine-tune such models on a system with only 4GB of VRAM"*[5].

V oblasti inštrukčného učenia na novom jazyku sa LoRA takisto široko využíva. Projekt `GemmAI (2024)` adaptoval anglickú 7-miliardovú verziu modelu `Gemma-7B` na prácu s inštrukciami v arabčine. Na tento účel bola vygenerovaná rozsiahla syntetická trénovacia množina arabských inštrukcií, skombinovaná s ľudsky anotovaným datasetom `InstAr-500k`. Model prešiel fázou monojazyčného *distillation* (prenosu znalostí z väčšieho modelu) a finálnym doladením na zmiešanom datasete. Počas fine-tuningu bol použitý `LoRA`, čo umožnilo znížiť počet trénovaných parametrov bez zníženia kvality. Ako autori uvádzajú: *"LoRA helped us reduce the number of trainable parameters, making the fine-tuning process more efficient without compromising performance."*[6]

Výsledný model `GemmArab` dosiahol vynikajúce výsledky v rôznych úlohách v arabčine, čo potvrdzuje praktickosť LoRA pre jazykovú adaptáciu LLM.

Okrem LoRA existujú aj iné PEFT-prístupy, ktoré sa v praxi používajú. *Adaptéry* (learnable adapters) sa vkladajú do vrstiev modelu ako malé medzičlánky s úzkym *bottleneck*-om (Houlsby et al., 2019). Aj tie sa využívali na jazykové prispôsobenie: napríklad tzv. *bilingual adapters* preukázateľne zlepšovali cross-lingválny prenos vo viacerých modeloch typu `BERT`.[7]

Ďalšou metódou je *prompt tuning*, teda učenie špeciálnych vopred definovaných tokenov (*promptov*), ktoré pri vstupe modelu navodia požadovaný jazykový výstup. Samotné učenie niekoľkých prompt-embeddingov však zvyčajne nestačí na zvládnutie úplne nového jazyka - je efektívne iba vtedy, ak model už má o danom jazyku určité znalosti. V praxi sa prompt- alebo prefix-tuning často používa ako doplnok k iným metódam (napr. na štylistické doladenie), ale pre samotnú jazykovú adaptáciu sú výrazne účinnejšie práve `LoRA` a *adaptéry*.

V skratke: v komunite sa najväčšej popularite tešia práve metódy ako `LoRA`, ktoré umožňujú rýchlo, lacno a efektívne pridať do modelu novú jazykovú schopnosť.

4.3 Problémy metódy LoRA pri jazykovej adaptácii

Napriek pôsobivým úspechom nie je `LoRA` univerzálnym riešením - viacero štúdií z rokov 2023–2024 poukázalo na jemné rozdiely a obmedzenia v porovnaní s plným fine-tuningom. V práci *LoRA vs Full Fine-tuning: An Illusion of Equivalence* (Shuttleworth et al., 2024) sa detailne analyzuje, ako LoRA mení štruktúru váhových matíc v LLM v porovnaní s úplným doladením modelu.[8]

Ukázalo sa, že pri použití LoRA vznikajú v maticiach modelu nové „vnikajúce“ singulárne komponenty - teda vektory v SVD rozklade, ktoré sa v plne doladenom modeli nevyskytujú. Inými slovami, aktualizácie s nízkym rankom pridávajú do váh nové smery, ktoré nekorešpondujú s pôvodným priestorom predtrénovaného modelu. Tieto tzv. *intruder dimensions* sa navyše často umiestňujú vysoko v spektrálnom poradí (majú veľké singulárne hodnoty), ale neodzrkadľujú predtým existujúce komponenty modelu.

Naopak, plný fine-tuning posúva singulárne hodnoty plynulejšie a zostáva v rámci pôvodného spektrálneho simplexu - bez náhlych výskokov nových hlavných komponentov. Zaujímavé je, že v rámci trénovacieho rozdelenia sa tieto rozdiely zvyčajne kvalitatívne neprejavia: LoRA model často dosiahne rovnakú presnosť ako plne doladený model.

Avšak mimo pôvodného trénovacieho rozdelenia sa rozdiely stávajú významné. Zistilo sa, že LLM po aplikácii LoRA horšie uchováva *mentálny model sveta*, ktorý si osvojila počas predtrénovania, a je menej robustná pri postupnom dolaďovaní na viacerých úlohách.

Jednoducho povedané, model s LoRA má vyššiu tendenciu „zabúdať“ alebo skresľovať niektoré pôvodné znalosti v porovnaní s modelom doladeným plne - aj keď dosahujú rovnakú presnosť na konkrétnej úlohe.

V kontexte jazykovej adaptácie to znamená potenciálne riziko: ak sa model doladí pomocou LoRA na nový jazyk, môže dôjsť k oslabeniu jeho schopností spracovávať texty v pôvodnom jazyku alebo k zníženiu generalizačnej schopnosti pri prenose na iné úlohy.

Dôležitou metrikou pri hodnotení účinnosti LoRA je tzv. *efektívny rank* váhových matíc adaptéra. Hoci LoRA explicitne obmedzuje rank prídavných matíc, výskum ukazuje, že aj pri zvýšení ranku r zostáva efektívny vplyv LoRA na váhy menší než pri plnom fine-tuningu.[8]

Inými slovami, pridanie väčšieho počtu parametrov (zväčšenie r) neznamená, že LoRA dokáže úplne replikovať voľné aktualizácie všetkých váh - model sa stále učí len v obmedzenom podpriestore. To obmedzuje schopnosť LoRA vyjadrovať veľmi komplexné transformácie, napríklad osvojovanie si novej gramatiky so zložitou štruktúrou.

Výskumníci dochádzajú k záveru, že LoRA a plný fine-tuning využívajú rozdielne časti priestoru riešení LLM.[8] Na dosiahnutie podobného riešenia ako pri plnom fine-tuningu je niekedy potrebné výrazne zvýšiť rank LoRA až po tzv. *stabilizačnú hodnotu*, čím sa však znižujú výhody z hľadiska efektivity.

Ďalším problémom sú špecifiká gradientného zostupu pri použití LoRA. V počiatočných krokoch učenia môže obmedzenie na nízky rank spôsobovať, že optimálny gradientový krok nie je realizovateľný - kvôli faktorizačnému tvaru matíc. Formálne, ak vyjadríme aktualizáciu váh ako $W \leftarrow W + \Delta W$, tak LoRA zavádza obmedzenie

$\Delta W = AB$, kde $A \in \mathbb{R}^{d \times r}$ a $B \in \mathbb{R}^{r \times d}$.

Na začiatku trénovania náhodná inicializácia A a B znamená, že aktualizácie prebiehajú nie v smere skutočného (anti)gradientu, ale len v rámci obmedzeného podpriestoru. To prispieva k vzniku spomínaných „vnikajúcich“ komponentov a môže spomaliť konvergenciu alebo dokonca naviesť model do iného lokálneho minima než pri plnom doladení.

Tento rozdiel sa najviac prejaví pri malých alebo homogénnych dátach - model s LoRA môže vykazovať výraznejšie preučenie. Napriek tomu platí, že s rastúcim počtom epoch LoRA modely často dosiahnu porovnateľnú výkonnosť na cieľovej úlohe ako modely s plným fine-tuningom,[8] hoci cesta k tomuto riešeniu bola odlišná.

Osobitnu pozornost si zasluhuju otázky preučenia a odolnosti modelu. Na jednej strane, znížený počet trénovateľných parametrov v `LoRA` znižuje riziko triviálneho preučenia pri malom datasete - model jednoducho nemá dostatočnú kapacitu na to, aby si „zapamätal“ všetky príklady. Na druhej strane však práca Shuttleworth et al. (2024) ukázala, že modely s LoRA majú nižšiu schopnosť tzv. *continual learning* - teda postupného učenia sa na viacerých dátových sadách.[8]

Takýto model rýchlejšie stráca všeobecné reprezentácie a zručnosti nadobudnuté v predchádzajúcich fázach trénovania. Tento efekt možno chápať ako formu *spektrálneho preučenia*: model dosiahne vysokú presnosť na prvej úlohe, ale za cenu posunutia spektra váhových matíc, čo následne sťažuje učenie na ďalších úlohách. V kontexte jazykovej adaptácie to znamená, že ak sa model s LoRA postupne adaptuje na viacero nových jazykov, kvalita môže degradovať rýchlejšie než pri plnom doladení (fine-tuning). Takéto scenáre si vyžadujú buď starostlivé plánovanie poradia adaptácií, alebo použitie pokročilejších metód.

Praktické obmedzenia LoRA sa dotýkajú aj fázy inferencie a správy adaptérov. Ak chceme používať jednu základnú LLM pre viacero jazykov alebo úloh, pričom každý má vlastný LoRA modul, vznikajú režijné náklady pri spracovaní požiadaviek. Ak LoRA adaptér nie je zlúčený s hlavnými váhami, generovanie textu vyžaduje dodatočné maticové operácie v každej vrstve - konkrétne aplikáciu A a B. Pre jeden adaptér je tento výpočet relatívne zanedbateľný, ale pri viacerých adaptéroch (napr. v multijazyčných alebo multitaskových scenároch) sa oneskorenia kumulujú.

V priemyselných aplikáciách, kde má jeden model súčasne obsluhovať mnohých používateľov s rôznymi prispôsobeniami, môže LoRA výrazne zvýšiť latenciu odpovedí. Ako sa uvádza v práci `FanLoRA`, štandardná LoRA môže pridávať až ~30% výpočtovej záťaže na každý krok generovania kvôli opakovaným maticovým násobeniam v každej vrstve.[4]

Okrem toho, ak sa adaptéry uchovávajú oddelene (nie sú zlúčené s modelom), rastie aj pamäťová náročnosť počas inferencie. Tento problém sa dá čiastočne riešiť zlúčením

adaptérov do základného modelu, no tým strácame možnosť dynamického prepínania medzi rôznymi zručnosťami. Táto dilema je obzvlášť aktuálna pri veľkých multijazyčných chatbot systémoch: uchovávať samostatný model pre každý jazyk je neefektívne, ale dynamické načítavanie viacerých LoRA modulov počas behu je pomalé.

4.3.1 Zhrnutie hlavných obmedzení základnej metódy LoRA pri jazykovej adaptácii

Zistené obmedzenia metódy `LoRA` pri prispôsobovaní modelu na nový jazyk možno zhrnúť do nasledujúcich bodov:

- **Spektrálne a singulárne skreslenia** – vznik nových hlavných komponentov vo váhových maticiach, ktoré sa líšia od pôvodných, čo môže zhoršovať uchovávanie starých znalostí modelu. [8]
- **Obmedzená vyjadrovacia schopnosť** – nízkorozmerné adaptácie nemusia byť dostatočné na vyjadrenie komplexných transformácií potrebných na plné osvojenie si nového jazyka, najmä ak ide o jazyk so velmi odlišnou štruktúrou. Výskumy ukazujú, že `LoRA`/adaptéry majú nižšiu kapacitu na učenie sa nových poznatkov v porovnaní s úplným trénovaním všetkých parametrov. [7]
- **Problémy s následným dolaďovaním (continual fine-tuning)** – model s LoRA má nižšiu odolnosť voči postupnému učeniu sa na viacerých úlohách, pričom môže výraznejšie zabúdať predtým nadobudnuté schopnosti. [8]
- **Zvýšená latencia a zložitosť inferencie** – pri súčasnom použití viacerých LoRA modulov (napr. pre rôzne jazyky) rastie latencia a zvyšuje sa technická zložitosť nasadenia modelu.
- **Výber optimálneho ranku** – je náročné vopred odhadnúť, aký rank r poskytne dostatočnú vyjadrovaciu silu bez zbytočného navýšenia parametrov. Príliš nízky r môže viesť k nedostatočnému naučeniu jazyka, zatiaľ čo príliš vysoký r zmenšuje rozdiel oproti plnému fine-tuningu a stráca výhody efektivity. Tento kompromis je dodnes aktívnou témou výskumu. [9]

4.4 Možné riešenia problémov LoRA a vylepšenia

4.4.1 XGBLoRA: zvyšovanie ranku iteratívne (gradientný boosting LoRA)

Jedným z perspektívnych smerov vývoja je iteratívne pridávanie LoRA-adaptérov s malým rankom, po vzore *gradientného boostingu* v klasickom strojovom učení. Tento

prístup bol navrhnutý Zhang et al. (2024) pod názvom `XGBLoRA` - *Extreme Gradient Boosting LoRA*.[2]

Základná myšlienka spočíva v tom, že namiesto učenia jednej dvojice matíc A a B s pevným rankom model postupne trénuje niekoľko LoRA modulov s minimálnym rankom (napr. $r = 1$), ktoré pôsobia ako „slabí žiaci“. V prvej iterácii sa trénuje adaptér, ktorý koriguje základné váhy v smere zníženia chyby. Následne sa jeho vplyv „zmrazí“ (alebo čiastočne zafixuje) a v ďalšej iterácii sa trénuje nový adaptér, ktorý sa snaží odstrániť zostávajúcu chybu modelu (reziduál). Tieto postupné nízkorozmerné aktualizácie sa sčítavajú a vytvárajú finálnu zmenu váh. Výsledná matica ΔW tak predstavuje súčet viacerých matíc nízkeho rádu, čo efektívne zvyšuje celkový rank modelu.

Autori `XGBLoRA` preukázali, že tento prístup umožňuje dosiahnuť kvalitu porovnateľnú s úplným fine-tuningom, a to s výrazne menším počtom trénovaných parametrov. V experimentoch na rôznych NLP úlohách `XGBLoRA` konzistentne prekonávala klasickú `LoRA` a blížila sa kvalite uplného doladenia.[2]

Každý jednotlivý adaptér má pritom extrémne malý rozsah (rank 1–2), vďaka čomu je trénovanie veľmi efektívne z hľadiska zdrojov. Teoreticky bolo dokázané, že postupné zvyšovanie ranku konverguje k optimálnemu riešeniu, čím sa prekonáva rozdiel medzi obmedzeným nízkym rankom a ideálnym gradientom.[2]

Intuitívne možno povedať, že `XGBLoRA` rieši problém voľby fixného r: model si *sám* „dorastie“ potrebný rank podľa potreby, až kým nebude chyba dostatočne malá. Zaujímavosťou je, že takéto zloženie viacerých jednoduchých adaptérov je menej náchylné na preučenie - každý komponent je príliš jednoduchý na to, aby sa preučil, a ich súhrn tvorí robustný ansámbl, podobne ako v prípade boosting stromov.

V kontexte jazykovej adaptácie je `XGBLoRA` mimoriadne atraktívna. Po prvé, nové jazykové znalosti možno zavádzať postupne: napríklad v prvej iterácii sa model naučí všeobecné znaky jazyka (napr. že slovenčina používa určitú abecedu a základnú slovnú zásobu), a v ďalších iteráciách sa doladia jemnejšie morfologické a syntaktické aspekty.

Po druhé, vďaka svojej výpočtovej úspornosti umožňuje `XGBLoRA` doladiť modely aj na bežných grafických kartách. Uvádza sa, že pomocou tejto metódy možno jemne doladiť veľký jazykový model aj na spotrebiteľskej GPU (napr. RTX 4090), a to bez straty kvality.

Treba poznamenať, že výsledný efektívny rank aktualizácie pri použití `XGBLoRA` je rovný: počet iterácií × rank každého adaptéra za predpokladu, že jednotlivé adaptéry sú navzájom ortogonálne.

V experimentoch sa ukázalo, že už aj pri sekvencii adaptérov s rankom 1 je možné dosiahnuť rovnakú výkonnosť ako pri jednom klasickom LoRA adaptéri s rankom 8–16.[10]

Tento prístup možno interpretovať ako formu *ansámblovania zručností*: každý mi-

niatúrny adaptér zachytáva určitý aspekt jazyka, zatiaľ čo ich kombinácia vytvára silnú, mnohorozmernú adaptáciu. Moderné výskumy ukazujú, že `XGBLoRA` skutočne prekonáva tzv. „ilúziu ekvivalencie“ - modely trénované týmto spôsobom sú svojimi vlastnosťami bližšie k úplne doladeným modelom než ku klasickej `LoRA`.

Z tohto dôvodu sa `XGBLoRA` v kontexte adaptácie LLM na nový jazyk javí ako jeden z kľúčových a mimoriadne sľubných nástrojov na zlepšenie efektivity aj kvality doladenia.

4.4.2 Inicializácia pomocou SVD a zmiešaný prístup so šumom

Ďalšia skupina riešení sa zameriava na odstránenie samotných príčin vzniku tzv. "vnikajúcich"(intruzívnych) komponentov pri metóde `LoRA`. Ako bolo spomenuté vyššie, jedným z dôvodov je náhodná inicializácia matíc A a B, kvôli ktorej začína učenie v náhodných, neštruktúrovaných smeroch. Vzniká tak myšlienka: inicializovať `LoRA`-adaptér na základe spektrálnej štruktúry pôvodných váh. V ideálnom prípade chceme začať učenie v tom istom podpriestore, v ktorom sa nachádzajú už naučené znalosti modelu, a len ich mierne korigovať.

Túto ideu realizuje metóda `OLoRA` (*Orthonormal LoRA*) od Büyükakyüz (2024).[11] `OLoRA` vykonáva rozklad váhovej matice W vrstvy pomocou ortonormálnej faktorizácie (QR-rozkladu): $W = QR$, kde Q je ortonormálna matica. Z nej sa vyberú prvé r stĺpce Q a prvé r riadky R – ich súčin $Q_r R_r$ tvorí nízkorozmernú aproximáciu pôvodnej matice. Tieto Q_r a R_r sa následne použijú ako počiatočné A a B v LoRA. Inými slovami, k matici W sa hneď na začiatku pridá $\Delta W \approx W_r$, ktorá priamo vychádza z existujúcich váh. Začiatočný adaptovaný model má teda váhy $W + \Delta W$, pričom ΔW zosilňuje najvýznamnejšie spektrálne zložky.

Na prvý pohľad sa to môže zdať kontraproduktívne (prečo duplikovať časť W?), ale práve takáto inicializácia zabezpečuje štart učenia v „dobrom“ podpriestore. Matice A a B sú ortogonálne a zosúladené s hlavnými komponentmi pôvodného modelu. Podľa výsledkov autora metódy vedie takáto inicializácia k rýchlejšej a stabilnejšej optimalizácii a k vyššej finálnej presnosti než pri náhodnej inicializácii LoRA.[11]

V experimentoch na rôznych jazykových úlohách `OLoRA` dosahovala vyššiu presnosť a rýchlejšiu konvergenciu v porovnaní s bežnou LoRA. Dôležité je, že `OLoRA` nepridáva žiadne nové parametre - ide len o spôsob inicializácie, ktorý je kompatibilný so všetkými variantmi LoRA (vrátane adaptívnych, boostingových atď.).

Pre účely jazykovej adaptácie je `OLoRA` atraktívna najmä tým, že zachováva spektrálne vlastnosti pôvodného modelu. Ak napríklad chceme doladiť model na slovenčinu, ktorý bol pôvodne trénovaný na angličtine, ortonormálna inicializácia bude korigovať váhy v už významných smeroch. To môže obmedziť zbytočný posun distribúcie a znížiť riziko straty znalostí angličtiny.

Na druhej strane sa nemožno úplne spoliehať na pôvodný priestor - nový jazyk môže vyžadovať nové dimenzie reprezentácie. Z tohto dôvodu je zaujímavý zmiešaný prístup: inicializovať adaptér v spektre W, no následne pridať malé náhodné rušenie alebo umožniť rast ranku v čase.

Myšlienka spočíva v kombinovaní výhod `OLoRA` a klasickej LoRA: začať učenie v ortonormálnom báze, ale umožniť modelu postupne z tohto priestoru „vyrásť“. V praxi to možno realizovať napríklad cez inicializáciu A, B pomocou zrezaného SVD (príbuzného QR) a následné pripojenie malého náhodného šumu, alebo dvojstupňovým učením: najprv zmraziť A a trénovať iba B, potom uvoľniť aj A - tento prístup čiastočne pripomína `LoRA-FA`, kde sa matica A drží fixne.[9]

Zároveň experimenty zo štúdie Shuttleworth et al. ukazujú, že ak sa počas trénovania fixuje náhodná matica A a učí sa len B, počet „intruzívnych“ komponentov v spektre váh dramaticky klesá.[8] To naznačuje, že kontrolovaním počiatočného bázu A možno výrazne redukovať spektrálne skreslenia.

Na záver, inicializácia LoRA adaptéra so zohľadnením SVD/QR štruktúry pôvodných váh je jednoduchý a efektívny spôsob, ako zrýchliť a zlepšiť adaptáciu na nový jazyk. Pridanie malého šumu alebo postupné rozmrazovanie parametrov umožní modelu vystúpiť z pôvodného jazyka do nového priestoru - napr. kvôli iným fonetickým javom alebo štruktúre viet.

Takýto hybridný prístup, hoci ešte nie je výslovne popísaný v literatúre, nadväzuje na poznatky posledných výskumov a v zásade predstavuje kombináciu `OLoRA` a regulačných techník. Predpokladá sa, že práve tento prístup umožní vytvoriť adaptáciu, ktorá minimálne naruší pôvodný model a zároveň plne zachytí špecifiká nového jazyka.

4.4.3 Adaptívne rozdelenie ranku (AdaLoRA a iné)

Ako už bolo uvedené, voľba ranku r pre `LoRA` je netriviálna úloha: rôzne vrstvy modelu môžu vyžadovať rôzne množstvo úprav. Metóda `Adaptive Low-Rank Adaptation (AdaLoRA)`, navrhnutá Zhang et al. (2023), rieši tento problém dynamickým prerozdeľovaním parametrového rozpočtu medzi vrstvami.[3]

Ako už bolo spomenuté v kapitole II, v `AdaLoRA` sa na začiatku stanoví celkový maximálny rozpočet ranku (napr. ekvivalent $r = 8$ na jednu vrstvu). Počas trénovania metóda vyhodnocuje dôležitosť aktualizácií v jednotlivých vrstvách a dokáže „orezať“ menej významné singulárne hodnoty z matíc AB v niektorých vrstvách. Takto uvoľnený rank sa následne prerozdelí do iných vrstiev, kde je chyba ešte vysoká a potrebné väčšie úpravy.

Technicky `AdaLoRA` parametrizuje aktualizácie cez SVD: počas trénovania sa počíta singulárny rozklad aktuálnej ΔW a najmenej významné komponenty sa odstránia, čím sa zníži efektívny rank v danej vrstve. Uvoľnené „sloty“ sa pridelia tam, kde sú najviac

potrebné. Tento prístup pripomína adaptívnu regularizáciu - model si sám určuje, kde potrebuje viac voľnosti na prispôsobenie.

`AdaLoRA` dosiahla v experimentoch lepšiu kvalitu než klasická LoRA s pevným rankom, najmä v podmienkach obmedzeného počtu parametrov. Napríklad pri obmedzení na 0,5,% trénovateľných parametrov prekonala `AdaLoRA` klasickú LoRA vo viacerých úlohách klasifikácie a generovania textu. To naznačuje, že metóda dokázala efektívne „skoncentrovať“ kapacitu modelu tam, kde to bolo kriticky dôležité.

V kontexte jazykovej adaptácie je to mimoriadne relevantné: model môže napríklad potrebovať viac ranku vo vrstvách blízko tokenizéru (na zachytenie ortografie nového jazyka), alebo v horných vrstvách (na uchopenie sémantických štruktúr). Jednotný r pre všetky vrstvy je v takom prípade neefektívny. `AdaLoRA` však sama identifikuje, kde sa nachádzajú „úzke miesta“.

Zaujímavosťou je, že v `AdaLoRA` sa používa regularizácia, ktorá podporuje ortogonálnosť aktualizácií (na potlačenie redundancie). To nadväzuje na princípy `OLoRA` - rozdiel je v tom, že `OLoRA` zavádza ortogonalitu hneď pri inicializácii, zatiaľ čo `AdaLoRA` ju presadzuje počas trénovania pomocou penalizácie.

V súvislosti s prekonávaním nedostatkov LoRA adaptívny prístup zvyšuje tzv. *parametrickú efektivitu* - teda dosiahnutie čo najvyššej kvality pri pevne danom počte parametrov. Model trénovaný pomocou `AdaLoRA` tak pri rovnakom rozpočte dosahuje vyššiu vyjadrovaciu silu než pri klasickej LoRA.

Okrem `AdaLoRA` vznikli aj ďalšie prístupy k riadeniu ranku:

- `LoRA-drop` (2024)[12] - po trénovaní analyzuje prínos jednotlivých častí adaptéra k výstupu a odstráni tie, ktoré sú nadbytočné (*pruning*). Ide teda o *a posteriori* verziu adaptivity - podobne ako AdaLoRA, ale aplikovanú až po doladení.

- `Delta-LoRA` (Sun et al., 2023) - rozširuje klasickú LoRA tým, že okrem matíc A a B postupne upravuje aj samotné váhy W (ako ΔW). Autori tvrdia, že nízkorozmerné matice nie sú vždy postačujúce a umožňujú obmedzené zásahy aj do hlavných váh, pričom zostávajú výpočtovo efektívnejší než plný fine-tuning.

- `RoSA` (Robust and Sparse Adaptation) (Ding et al., 2024)[9] - kombinuje nízkorozmerné a sparsné (riedke) aktualizácie. Myšlienka je taká, že niektoré kritické zmeny možno lokalizovať v konkrétnych váhach (ktoré zachytí sparsné komponenty), zatiaľ čo zvyšné zmeny majú globálnejší nízkorozmerný charakter.

Zhrnutie: Adaptívne riadenie ranku - či už počas trénovania (`AdaLoRA`), alebo po ňom (`LoRA-drop`) - predstavuje výkonný nástroj na zlepšenie LoRA. `AdaLoRA` rieši problém pevného ranku tým, že modelu umožní rozhodnúť, kde potrebuje viac parametrov a kde menej. Pre komplexnú úlohu ako je osvojenie si nového jazyka je takáto

flexibilita mimoriadne cenná: rôzne jazykové aspekty (fonetika, morfológia, syntax) vyžadujú rôznu úroveň zložitosti modelu.

Uplatnenie AdaLoRA a príbuzných techník by preto malo viesť k plnšiemu osvojeniu si nového jazyka modelom - s minimálnym počtom parametrov - a zároveň znížiť riziko, že model "prehliadne"niektoré jazykové vlastnosti len preto, že na ne nemal kapacitu.

4.4.4 Zlučovanie adaptérov a kompozičné metódy

Ďalším smerom v zlepšovaní LoRA je *kompozícia viacerých adaptérov* za účelom dosiahnutia komplexných zručností. Ak má model zvládať viacero jazykov alebo úloh, prirodzené vzniká myšlienka naučiť jednotlivé adaptácie samostatne a následne ich *zjednotiť*.

Najjednoduchší prístup je tzv. *zlúčenie LoRA modulov*, napríklad prostým sčítaním alebo spriemerovaním delta-váh. Avšak priame sčítanie nezávisle trénovaných adaptérov nemusí viesť k optimálnemu výsledku a môže spôsobiť interferenciu medzi ich účinkami.

Pokročilejší prístup navrhli Zhao et al. (2023) pod názvom AdaMergeX - *Adaptive Adapter Merging*.[13] Táto metóda rozdeľuje schopnosť LLM riešiť úlohu v konkrétnom jazyku na dve zložky:

1. *Schopnosť riešiť úlohu* (task ability)

2. *Schopnosť pracovať s daným jazykom* (language ability)

Napríklad: adaptér trénovaný na úlohe (napr. otázky a odpovede) v angličtine predstavuje komponent (a), zatiaľ čo adaptér trénovaný na jednoduchej referenčnej úlohe v slovenčine predstavuje komponent (b) - teda samotnú znalosť jazyka.

AdaMergeX potom vykoná štruktúrne zlúčenie: vezme jazykový adaptér (b) a úlohový adaptér (a) a spojí ich tak, aby vznikol nový adaptér riešiaci danú úlohu v novom jazyku. Na rozdiel od jednoduchého sčítania váh využíva špeciálny algoritmus, ktorý zohľadňuje štruktúru matíc a prispôsobuje ich vzájomne - ide teda o *štruktúrne adaptívne zlučovanie*.

V experimentoch AdaMergeX prekonal všetky naivné metódy prenosu a umožnil efektívny *zero-shot* transfer do nového jazyka bez toho, aby bol model trénovaný na konkrétnej kombinácii úloha–jazyk.

V kontexte LoRA to možno aplikovať napríklad tak, že model doladíme na anglický QA pomocou jedného LoRA modulu a osobitne na slovenský preklad pomocou druhého. Zlúčením týchto modulov dostaneme model schopný odpovedať na otázky v slovenčine.

Takýto prístup je obzvlášť užitočný, keď v cieľovom jazyku existuje málo údajov - problém sa dá rozdeliť na dve časti: znalosť jazyka a schopnosť riešiť úlohu. Pri

jazykovej adaptácii môžeme jazyk chápať ako špecifickú „úlohu“, pričom merging sa stáva formou multilingválnej schopnosti.

Zaujímavý je aj koncept `AdapterFusion` (Pfeiffer et al., 2021), kde sa niekoľko vopred natrénovaných adaptérov kombinuje pomocou meta-modelu, ktorý sa špeciálne naučí ich optimálne spájať.

V kontexte adaptácie LLM na nový jazyk má zlučovanie adaptérov potenciál v nasledujúcich prípadoch:

- **(a)** Zachovať pôvodné jazykové zručnosti
- **(b)** Pridať viacero nových jazykov bez opätovného trénovania
- **(c)** Kombinovať jazykové a úlohové adaptácie

Prakticky je možné trénovať samostatné `LoRA` adaptéry pre každý jazyk a aktivovať ich podľa potreby - tento prístup už dnes využívajú niektoré multilingválne systémy. Alternatívne možno použiť `AdaMergeX` na získanie jedného zlúčeného adaptéra, ktorý pokrýva viacero jazykových schopností.

Tieto prístupy sú zatiaľ prevažne experimentálne, no vykazujú sľubné výsledky, najmä pokiaľ ide o prenos medzi jazykmi bez straty pôvodných schopností modelu. Pre nízkozdrojové jazyky je kombinácia adaptérov veľkou výhodou: umožňuje „požičať si“ znalosti z bohatších jazykov (napr. prostredníctvom spoločného task-adaptéra) a tým vyrovnať nedostatok údajov v cieľovom jazyku.

4.4.5 Tréning tokenizéra a rozšírenie slovníka

Posledná skupina prístupov sa týka práce so vstupnými a výstupnými reprezentáciami modelu - teda s tokenizáciou a embeddingmi. Veľkou prekážkou pri pridávaní nového jazyka býva nekompatibilita abecedy alebo slovnej zásoby. Ak pôvodný model nikdy nevidel znaky určitého typu (napr. hieroglyfy alebo znaky s diakritikou), jeho vstupná embedding matica im nepriraďuje žiadne vhodné vektorové reprezentácie.

Jedným z riešení je **rozšírenie slovníka**: pridanie nových tokenov pre špecifické znaky alebo slová daného jazyka a natrénovanie ich embeddingov na textovom korpuse. Tento prístup bol úspešne demonštrovaný napr. v projekte `DictaLM2.0` pre hebrejčinu, kde bolo pridaných 1000 nových tokenov a natrénovaných v priebehu jednej epochy na zmiešanom anglicko-hebrejskom datasete. Už po tejto fáze došlo k výraznému zlepšeniu výkonu na hebrejčine.[14]

Dôležité je, že zvyšné váhy modelu sa v tomto štádiu nemenili, čo zachovalo schopnosti v angličtine. Až v druhej fáze sa uskutočnilo doladenie zvyšku modelu pomocou LoRA.[14]

Tento dvojfázový prístup - tzv. *embedding surgery* (doslova „transplantácia embeddingov“) - umožňuje minimalizovať deštrukciu predchádzajúcich znalostí. Výskum Artetxe et al. (2020) navyše ukázal, že aj bez akéhokoľvek doladenia vnútra modelu je možné dosiahnuť rozumné výsledky len učením embeddingov nového jazyka na základe spoločných čŕt so známymi jazykmi.

V nedávnej práci `Franken-Adapter` (ICML 2025) autori zdôrazňujú, že „customizing tokenizers is critical for enhancing language adaptation“ a uvádzajú až 20,% zlepšenie kvality pri 96 jazykoch pomocou špeciálneho adaptéru a doladeného tokenizéra, pričom zníženie výkonu na angličtine nepresiahlo 1,% straty.

Tieto výsledky naznačujú, že správna práca s tokenmi - ich doplnenie a trénovanie ich embeddingov - môže významne zvýšiť efektivitu jazykovej adaptácie.

Pokiaľ ide o výstupné reprezentácie, možno samostatne doladiť aj **finálnu lineárnu vrstvu** (`LM Head`) pre nový jazyk, najmä ak boli pridané nové tokeny. V prípade DictaLM bola táto fáza označovaná ako *LM-Head Calibration* - kalibrácia výstupnej vrstvy po rozšírení slovníka.[14]

Tento krok pomáha zlepšiť generovanie textu v novom jazyku bez toho, aby sa zhoršil výkon v starom. Celý postup možno chápať ako to, že sa modelu dajú „správne stavebné kocky“ - teda zmysluplné vstupno-výstupné reprezentácie - a až potom sa doladia interné váhy.

Praktické odporúčania pre adaptáciu cez tokeny zahŕňajú:

- **Rozšírenie tokenizéra** tak, aby pokrýval špecifické znaky a výrazy nového jazyka.
- **Tréning nových embeddingov** (aj výstupných váh) na veľkom korpuse nového jazyka, pričom ostatné parametre modelu ostanú zamrznuté.
- **Následné využitie LoRA alebo adaptérov**, keď má model už pevný lexikálny základ.
- **V prípade potreby - použitie špeciálnych markerov alebo promptov** na signalizáciu jazyka, aby model vedel, v akom kontexte odpovedať.

V skratke tieto opatrenia umožňujú LLM efektívne pracovať s novými tokenmi a slovami. Aj keď interná architektúra transformera môže zostať do značnej miery „anglocentrická“, nové embeddingy a výstupné vrstvy pomôžu premietať naučené schopnosti na nový jazyk.

4.5 Metodika dolaďovania LLM pre slovenský jazyk na báze *GPT-Neo-125M*

V tejto časti sa opisuje proces adaptácie veľkého jazykového modelu (LLM) pre slovenský jazyk na základe predtrénovaného modelu *GPT-Neo-125M*. Adaptácia sa realizovala metódou dolaďovania na korpuse otázok a odpovedí v slovenčine s použitím optimalizátora **AdamW** (koeficient váhovej regularizácie weight_decay = 0,05; rýchlosť učenia learning rate = 3×10^{-5}). Dolaďovanie prebiehalo pri zmrazených pôvodných váhach modelu a využívalo zapojenie nízkorankových adaptérov.

Nižšie je uvedené odôvodnenie voľby modelu a hyperparametrov, popis cieľového dátového korpusu a porovnanie troch prístupov dolaďovania:

1. **LoRA** v implementácii knižnice *PEFT*;
2. **Vlastná verzia LoRA** (chápaná ako špeciálny prípad XGBLoRA bez iteratívnych aktualizácií);
3. **Navrhovaná metoda XGBLoRA** so sekvenčným pridávaním adaptérov.

Treba poznamenať, že napriek odporúčaniu pôvodnej práce o XGBLoRA používať rank $r = 1$ sa pri zvolenej úlohe jazykovej adaptácie nasadili adaptéry s vyšším rankom, aby sa modelu poskytla dostatočná flexibilita.

4.5.1 Výber modelu a hyperparametrov

Ako základný model na dolaďovanie bol zvolený *GPT-Neo-125M* obsahujúci približne ~ 125 miliónov parametrov. Ide o transformer-generátor predtrénovaný na rozsiahlych textových korpusoch; zvolený bol z dôvodu dostupnosti a primeranej veľkosti pre experimentálnu overenie adaptačných metód. Hoci *GPT-Neo-125M* zaostáva rozsahom za súčasnými viacmiliardovými LLM, jeho veľkosť umožnila realizovať sériu experimentov v obmedzených výpočtových podmienkach a zároveň zachovať relevantnosť výsledkov pre úlohy porozumenia slovenskej textovej správy.

Na dolaďovanie sa použil optimalizátor **AdamW**, ktorý sa osvedčil pri jemnom nastavovaní transformerov. Koeficient *weight decay* bol nastavený na 0,05, čím sa zabezpečila mierna L_2-regularizácia brániaca nadmernému prispôsobeniu adaptérov trénovacej množine. Rýchlosť učenia bola zvolená 3×10^{-5} na základe predbežných testov a odporúčaných hodnôt pre modely podobnej veľkosti; takto zvolený krok bol dostatočne malý na stabilné znižovanie funkcie straty bez prudkých výkyvov, no zároveň dostatočne veľký na konvergenciu v rozumnom počte epoch. Tréning prebiehal na dávkach pevnej veľkosti (určenej experimentálne s ohľadom na kapacitu GPU pamäte) a trval

najviac 5 epoch, po ktorých sa validačné metriky stabilizovali. Všetky kľúčové hyperparametre - optimalizátor, rýchlosť učenia, regularizácia, počet epoch - boli zjednotené pre všetky porovnávané adaptačné metódy, aby bolo zabezpečené korektné porovnanie výsledkov.

4.5.2 Korpus `TUKE-DeutscheTelekom/SKQuAD`

Na špecializáciu modelu na slovenský jazyk v úlohe porozumenia textu bol použitý korpus **SKQuAD**, vyvinutý v spolupráci s Technickou univerzitou v Košiciach (TUKE) a výskumného oddelenia Deutsche Telekom. Ide o prvý veľkorozsahový dataset typu *otázka–odpoveď* v slovenčine, štruktúrou zodpovedajúci anglickému *SQuAD*. Obsahuje približne 91 000 dvojíc „otázka – odpoveď“. Súbor je rozdelený na trénovaciu časť (≈ 81,6 tis. otázok) a validačnú časť (≈ 9,6 tis. otázok). Pre každú otázku je k dispozícii kontextový úryvok z článku a jedna či viac správnych odpovedí (zvyčajne krátke textové fragmenty). Dataset bol ručne anotovaný priamo v slovenčine, čo vylučuje artefakty strojového prekladu a robí úlohu realistickejšou pre dané jazykové prostredie. Pri dolaďovaní *GPT-Neo-125M* na SKQuAD model dostával na vstup otázku a prislúchajúci kontext a mal generovať správnu odpoveď. Kvalita odpovedí sa hodnotila metrikami *Exact Match* (EM) a *F1* na validačnej vzorke, v súlade so štandardom úloh strojového čítania. EM udáva podiel odpovedí, ktoré sa presne zhodujú s niektorou z referenčných odpovedí, zatiaľ čo F1 zachytáva čiastočné zhodovanie (prekryvanie podreťazcov odpovede a správnej frázy). Metrika F1 bola zvolená ako hlavné kritérium porovnania kvality rôznych adaptačných metód na SKQuAD, keďže je menej prísna a informatívnejšia v prípadoch synonymie alebo parciálnych zhôd.

4.5.3 Režimy dolaďovania: PEFT LoRA, vlastná LoRA a XGB-LoRA

V experimentálnej časti boli implementované tri režimy dolaďovania modelu s využitím nízkorankových adaptérov.

PEFT LoRA. V prvom režime sa uplatnil štandardný prístup nízkorankovej adaptácie prostredníctvom knižnice *PEFT* (Low-Rank Adaptation) od HuggingFace. Do každého modulu self-attention modelu *GPT-Neo-125M* boli vložené trénovateľné matice-adaptéry malého ranku (LoRA), ktoré jemne korigovali pôvodné transformácie. Všetky pôvodné váhy transformera zostali zmrazené; aktualizovali sa iba parametre vložených LoRA-adaptérov (a prípadne finálnej lineárnej hlavy, ak si to úloha vyžadovala). Pre slovensky korpus SKQuAD bol otestovaný rank adaptérov $r = 32$ a $r = 8$. Adaptéry s $r = 8$ obsahujú menej parametrov než $r = 32$, no modelujú menej zložité korekcie.

Výsledky (pozri kapitolu 5) ukázali, že zníženie ranku z 32 na 8 prakticky nezhoršilo kvalitu na SKQuAD, čo svedčí o dostatočnosti malého počtu adaptačných parametrov pre danú úlohu.

Vlastná implementácia LoRA (špeciálny prípad XGBLoRA). V druhom režime sa použil adaptér vyvinutý v rámci práce, funkčne ekvivalentný s LoRA, no implementovaný cez mechanizmus iteratívneho pridávania adaptérov ako pri XGBLoRA s obmedzením na jednu iteráciu. Inými slovami, v prvej (a jedinej) iterácii sa natrénuje nízkorankový adaptér a proces sa zastaví bez ďalšieho pridávania adaptérov. Tento špeciálny prípad zodpovedá bežnému LoRA, avšak v našom vlastnom kóde, čo uľahčilo priame porovnanie s viacnásobným aktualizovaním v XGBLoRA. Pri nezmenených podmienkach (hyperparametre, inicializácia) poskytlo jednorazové pridanie adaptéra výsledky porovnateľné s knižničným PEFT LoRA. Rank adaptérov ($r = 8$ alebo $r = 32$) bol zhodný s ostatnými experimentmi, pričom nedochádzalo k priebežným *merge* operáciám, lebo adaptér bol jediný a zlúčil sa s bázovými váhami až po ukončení tréningu.

XGBLoRA s iteratívnym pridávaním adaptérov. Tretí režim predstavuje navrhovanú metódu **Extreme Gradient Boosting Low-Rank Adaptation (XGBLoRA)**, prispôsobenú našej úlohe. Dolaďovanie prebieha iteratívne: po každej epoche sa natrénovaný nízkorankový adaptér *zmerguje* s váhami základného modelu, následne sa inicializuje nový (spravidla nulový) adaptér a v nasledujúcej epoche sa učí na zostatkových chybách. Proces pripomína gradientný boosting, v ktorom každý nový „slabý účastník“ (tu adaptér s malým rankom) vylepšuje celkovú kvalitu ansámblu. V originálnej práci o XGBLoRA sa odporúča použiť minimálny rank $r = 1$; v kontexte našej úlohy sa však tento variant ukázal ako nedostatočný, pretože adaptácia na nový jazyk a komplexnú úlohu strojového čítania vyžaduje väčšiu kapacitu v každom kroku. Zvolili sme teda kompromisný prístup a v rôznych experimentoch sme pridávali adaptéry s rankom $r = 8$ (v niektorých prípadoch $r = 32$). Tým sa zvýšil počet trénovateľných parametrov v každom „kroku boostingu“, čo umožnilo lepšie zachytiť zložité jazykové vzory. Celkový počet iterácií (adaptérov) sa menil: testovali sa varianty s 5 aj s 32 iteráciami. Po každej iterácii sa model vyhodnotil na validačnej množine a prírastok kvality sa zaznamenal. Implementačne bol cyklus rozšírený tak, že na konci každej epochy došlo k zlúčeniu váh aktuálneho adaptéra s hlavným modelom a nasledovala nová epocha s novým adaptérom. Vďaka nízkej dimenzii jednotlivých modulov nedochádzalo k prudkému preučeniu a model sa zlepšoval postupne.

Zjednotené podmienky. Všetky tri režimy (*PEFT LoRA*, jednorazová *LoRA*, *XGBLoRA*) využívali identické východiskové nastavenia: rovnaký predtrénovaný checkpoint

GPT-Neo-125M, rovnaký trénovací korpus *SKQuAD* a tie isté hyperparametre optimalizácie. Takto bolo možné korektne porovnať vplyv samotnej adaptačnej metódy na výslednú kvalitu. Kvantitatívne výsledky a analýza tréningu pre každý prístup sú uvedené v kapitole 5.

Kapitola 5

Analýza výsledkov a chýb

V tejto kapitole je vykonaná analýza dosiahnutých výsledkov experimentov a identifikovaných chýb pri adaptácii veľkých jazykových modelov na anglický a slovenský dataset. Na základe experimentálnych dát sú rozobraté možné príčiny rozdielov vo výkonnosti rôznych metód a modelov, a navrhnuté potenciálne riešenia na zlepšenie adaptácie.

5.1 Prehľad výsledkov experimentov

Všetky uskutočnené experimenty a ich základné výsledky sú zosumarizované v Tabuľke 5.1. Tabuľka uvádza číselné označenie experimentu, použitý model a adaptačnú metódu, dataset s konkrétnou úlohou a dosiahnutý výkon modelu na testovacích dátach. Je pokrytý anglický klasifikačný benchmark GLUE (experimenty 1-3, 16-17) aj slovenská úloha otázka-odpoveď (SKQuAD, experimenty 4-15). Pre prehľadnosť boli dva dodatočné experimenty (č. 16 a 17) zamerané na varianty XGBLoRA vynechané z hlavného porovnania a slúžia len na doplňujúcu ilustráciu.

Z Tabuľky 5.1 možno vycitat, že vo väčšine prípadov dosahujú metódy XGBLoRA vyššiu presnosť než základná metóda LoRA. Najvýraznejší rozdiel je pozorovaný pri modeli BERT na slovenskom datasete (experiment č. 12 vs. 13-14), kde XGBLoRA prevyšuje tradičnú LoRA o približne 3 percentuálne body (F1 okolo 54.4 % oproti 51.6% a 51.72%). Na anglickom klasifikačnom benchmarku GLUE dosiahla XGBLoRA takisto najlepší výsledok (experiment č. 3: Acc 81.3 %), čo je zlepšenie o 1.6% oproti bežnej LoRA (79.7 %). V prípade modelu GPT-Neo na úlohe otázka-odpoveď v slovenčine bol prínos XGBLoRA menej výrazný, no konfigurácia s viacerými iteráciami (32 upd) a nižším rankom ($r = 8$) dosiahla najvyššie F1 = 39.6 %, čím mierne prekonala všetky varianty LoRA (najviac 37.2 %). Celkovo výsledky potvrdzujú, že prístup postupného zvyšovania ranku pomocou XGBLoRA dokáže zlepšiť výkon modelov na rôznych úlohách a jazykoch, hoci veľkosť tohto zlepšenia závisí od konkrétneho modelu a povahy úlohy.

Č.	**Model**	**Metóda**	**Dataset / úloha**	**Výsledok**
01	RoBERTa-base (EN)	PEFT LoRA	GLUE, klasifikácia	Acc = 78.4 %
02	RoBERTa-base (EN)	LoRA ($r = 8$)	GLUE, klasifikácia	Acc = 79.7 %
03	RoBERTa-base (EN)	XGBLoRA (6 × rank=8)	GLUE, klasifikácia	Acc = 81.3 %
04	GPT-Neo-125M	LoRA ($r = 32$, PEFT)	SKQuAD, QA	F1 = 0.37.07 %
05	GPT-Neo-125M	LoRA ($r = 8$, PEFT)	SKQuAD, QA	F1 = 37.19 %
06	GPT-Neo-125M	LoRA ($r = 32$, bez SVD)	SKQuAD, QA	F1 = 33.97 %
07	GPT-Neo-125M	LoRA ($r = 32$, SVD init)	SKQuAD, QA	F1 = 32.20 %
08	GPT-Neo-125M	XGBLoRA (5 upd, $r = 32$)	SKQuAD, QA	F1 = 35.78 %
09	GPT-Neo-125M	XGBLoRA (32 upd, $r = 8$)	SKQuAD, QA	F1 = 39.60 %
10	GPT-Neo-125M	XGBLoRA (32 upd, $r = 32$)	SKQuAD, QA	F1 = 36.59 %
11	GPT-Neo-125M	XGBLoRA (FP32, 5 upd, $r = 32$)	SKQuAD, QA	F1 = 35.78 %
12	BERT-base-uncased	XGBLoRA (5 × rank=32)	SKQuAD, QA	F1 = 54.42 %
13	BERT-base-uncased	LoRA ($r = 32$)	SKQuAD, QA	F1 = 51.60 %
14	BERT-base-uncased	PEFT LoRA (r = 32)	SKQuAD, QA	F1 = 51.72 %
15	BERT-base-uncased	Plný finetuning	SKQuAD, QA	F1 = 62.71 %
16	RoBERTa-base (EN)	XGBLoRA (10 upd, $r = 8$)	GLUE, klasifikácia	Acc = 0.8091 %
17	RoBERTa-base (EN)	XGBLoRA (upd 1/3, $r = 8$)	GLUE, klasifikácia	Acc = 0.7656 %

Tabuľka 5.1: Prehľad uskutočnených experimentov a ich výsledkov.

Pre zjednodušenie, aby sme sa vyhli malým číslam, F1 je uvedená v percentách. Číselné označenie experimentu zodpovedá textu práce. Metódy: **LoRA** - nízkorozmerná adaptácia s rankom r (v zátvorke sú uvedené prípadné úpravy, napr. SVD init znamená inicializáciu váh adaptéra cez SVD); **XGBLoRA** - adaptácia metódou Extreme Gradient Boosting LoRA (v zátvorke počet boosting iterácií a rank); **Plný finetuning** - doladenie všetkých parametrov modelu.

5.2 Výsledky na anglickom datasete GLUE

Na anglickom klasifikačnom sete GLUE boli experimenty realizované s modelom RoBERTa-base. Z tabuľky výsledkov vyplýva, že všetky testované metódy adaptácie (LoRA aj XGBLoRA) dosiahli porovnateľne vysokú presnosť, pričom najlepší variant (XGB-LoRA so šiestimi iteráciami a rankom 8) dosiahol presnosť 81.3 %. Štandardná LoRA (rank 8) zaostala len mierne (79.7 %) a aj úplne základná implementácia PEFT LoRA dosiahla 78.4 %. Rozdiel medzi najlepším a najslabším variantom predstavuje približne 3 percentuálne body, čo naznačuje, že v prípade robustného anglického benchmarku s väčším množstvom dát nemajú nízkorozmerné adaptačné metódy výrazný problém dosiahnuť vysoký výkon. Dva dodatočné experimenty (č. 16 a 17) skúmali vplyv frekvencie spájania adaptérov v rámci XGBLoRA. Experiment č. 16 zvýšil počet boosting iterácií na 10, čo prinieslo výslednú presnosť 80.9 %, teda takmer na úrovni variantu so 6 iteráciami. Experiment č. 17 skúšal alternatívnu stratégiu s menej častým spájaním (adaptér bol zlúčený iba raz po odtrénovaní jednej tretiny epoch), výsledkom čoho bola mierne nižšia presnosť 76.6 %. Tieto výsledky naznačujú, že pôvodná stratégia (spájanie LoRA adaptéra po každej epoche, celkovo 6-krát) bola pre XGBLoRA na GLUE už blízko optimálnej. Príliš veľké zvyšovanie počtu iterácií neprinieslo významné zlepšenie a príliš zriedkavé spájanie naopak výkon zhoršilo.

5.3 Výsledky na slovenskom datasete SKQuAD

Pri adaptácii modelov na slovenský dataset otázok a odpovedí (SKQuAD) sa prejavili väčšie rozdiely medzi jednotlivými metódami. V experimentoch s modelom *GPT-Neo-125M* bolo dosiahnuté maximálne F1 = 39.6 % (XGBLoRA s 32 iteráciami a $r = 8$). Klasická LoRA pritom v najlepšom prípade (s vyšším rankom $r = 32$) dosiahla F1 okolo 37.1 %, pričom použitie nižšieho ranku $r = 8$ v kombinácii s viacerými boosting krokmi sa ukázalo efektívnejšie než jednoduché navýšenie ranku bez boostingu. Zaujímavým zistením je výsledok experimentu 6 v Tabuľke 5.1 kde sa SVD-inicializácia síce použila, no tento variant napriek tomu dosiahol najhoršie výsledky - horšie než experiment 5, v ktorom SVD nebola nepoužitá a adaptéry boli nainicializované náhodne. Experiment 6 dosiahol najnižšiu hodnotu F1 = 32.20%, čo však pravdepodobne súvisí so štatistickou odchýlkou a nie s negatívnym vplyvom SVD. Predchádzajúce testy totiž ukázali, že SVD-inicializácia prináša malý, ale konzistentný prínos k metrikám, preto je zahrnutá vo všetkých ostatných scenároch s mojou implementáciou LoRA/XGBLoRA. Celkovo však aj najlepšia konfigurácia modelu GPT-Neo výrazne zaostávala za výsledkami dosiahnutými modelom BERT na rovnakej úlohe. V experimentoch s *BERT-base-uncased* (multilingválnym modelom) na SKQuAD dosiahla metóda uplného doladenia modelu jednoznačne najvyššiu kvalitu odpovedí (F1 = 62.7 %, $EM \approx 37$ %). To značí, že

pri dostatočnom prispôsobení všetkých parametrov vie BERT efektívne využiť svoje predtréningové znalosti pre porozumenie slovenskému textu. Z parameter-efektívnych metód dosiahla najlepší výkon XGBLoRA (F1 = 54.4 %), ktorá tak prekonala klasickú LoRA (51.6 %) aj variant s PEFT LoRA adaptérmi (51.7 %). XGBLoRA však vďaka postupnému zvyšovaniu ranku dokázala získať výraznejšie zlepšenie, hoci stále nedosahovala úroveň plne doladeného modelu (zaostávala približne o 8 percentuálnych bodov v metrike F1). Tento rozdiel poukazuje na limitácie nízkorozmerných adaptácií pri náročnejších úlohách: aj keď významne uľahčujú a zrýchľujú proces trénovania, určitá časť modelovej kapacity zostáva nevyužitá, čo sa prejaví v slabšom výkone v porovnaní s plným doladením.

5.4 Analýza chýb

Hoci metóda XGBLoRA vo všeobecnosti zlepšila výsledky oproti základnej LoRA, podrobná analýza odhalila niekoľko oblastí, kde tieto adaptácie stále zaostávali za plne doladeným modelom. Pri úlohe otázka-odpoveď bolo pozorované, že modely trénované pomocou LoRA (vrátane XGBLoRA) častejšie vracali neúplné alebo menej presné odpovede. Napríklad pri niektorých otázkach síce model identifikoval správnu pasáž v kontexte, ale nedokázal extrahovať celú požadovanú odpoveď alebo pridal nadbytočné slová. Tieto chyby naznačujú, že obmedzená kapacita adaptéra nedokáže vždy dostatočne postihnúť komplexné jazykové vzťahy potrebné na plné porozumenie textu. Ako jedna z možných príčin sa javí tzv. *spektrálne skreslenie*, ktoré vzniká pri nízkorozmerných adaptáciách - model sa môže uberať odlišným smerom v priestore svojich reprezentácií než pri plnom doladení. To môže viesť k čiastočnej strate pôvodných znalostí alebo k neoptimálnemu využitiu parametrov, čo sa následne prejaví chybami v predikciách. Tiež platí, že ak má adaptér príliš malý rank, nemusí zvládnuť vyjadriť všetky dôležité vzťahy v dátach, najmä ak ide o jazyk so zložitou morfológiou a volnym slovosledom, akým je slovenčina. Možným riešením týchto problémov je ďalšie zvýšenie kapacity adaptácie alebo kombinácia viacerých prístupov. Napríklad, ak by to výpočtové zdroje dovolili, možno uvažovať o zvýšení ranku nad hodnoty testované v práci, prípadne o využití adaptéra vo viacerých vrstvách s rôznymi rankami. Inou cestou je čiastočné doladenie kritických vrstiev modelu popri použití LoRA - tým by sa mohla zachovať výhoda nižšej trénovacej náročnosti, ale model by získal väčšiu pružnosť v adaptácii na nový jazyk. Perspektívnym smerom výskumu je tiež skúmanie metód ako LoRA-XS či OLoRA, ktoré sa snažia zmierniť spektrálne skreslenie pomocou rozkladu váh alebo ortogonálnych regularizácií. V neposlednom rade, pre úlohy vyžadujúce hlboké porozumenie textu (ako QA) môže byť prínosné zamerať sa aj na kvalitatívnu analýzu chýb modelu a následne cielene upraviť trénovacie dáta

alebo architektúru adaptéra tak, aby sa eliminovali konkrétne zlyhania (napr. doplnením ďalších príkladov otázok, na ktorých model zlyháva, alebo zavedením mechanizmu verifikácie odpovede). Napokon, výsledky experimentov potvrdzujú očakávanie, že plné doladenie veľkého modelu prináša najvyšší možný výkon za cenu najväčších nárokov na zdroje. Metódy ako LoRA a XGBLoRA predstavujú kompromis - výrazne znižujú počet trénovaných parametrov a pamäťovú záťaž, pričom dosahujú výkon blízky optimálnemu najmä na menej komplexných úlohách. Pri náročnejších scenároch alebo pri postupnej adaptácii modelu na viaceré úlohy však môžu nastať isté problémy (napr. zabúdanie predchádzajúcich znalostí či nedostatočné naučenie nových vzorov), ktoré je potrebné ďalej skúmať. Napriek týmto obmedzeniam sú nízkorozmerné adaptačné metódy mimoriadne cenným nástrojom, najmä pre aplikácie v prostredí s obmedzenými výpočtovými zdrojmi alebo pri práci s jazykmi, pre ktoré nie sú k dispozícii rozsiahle predtréningové dáta.

Kapitola 6

Záver

V tejto záverečnej kapitole stručne zhŕnieme dosiahnuté výsledky bakalárskej práce, zhodnotíme splnenie stanovených cieľov a naznačíme možnosti ďalšieho rozvoja navrhnutého riešenia.

Analýza experimentálnych údajov odhalila niekoľko kľúčových prínosov navrhnutého modelu:

- **Zlepšenie presnosti:** *XGBLoRA* dosiahol v priemere o niekoľko percent vyššiu klasifikačnú presnosť v porovnaní s klasickým *LoRA* modelom, najmä na zložitých dátových súboroch s veľkou variabilitou.
- **Lepšia robustnosť:** Model preukázal lepšiu odolnosť voči šumu a nadmernému prispôsobeniu trénovacím údajom vďaka adaptívnemu modulu *LoRA*, ktorý umožnil účinnú reguláciu zložitosti modelu.
- **Zníženie výpočtovej náročnosti:** Navyše sme zaznamenali, že začlenenie mechanizmu *LoRA* umožnilo znížiť množstvo potrebných trénovacích parametrov, čo prinieslo mierne zrýchlenie trénovacieho procesu pri zachovaní celkovej výkonnosti.

Experimenty preukázali, že navrhnutý model *XGBLoRA* efektívne kombinuje výhody gradientného boostingového stroja a adaptívnej metódy nízkorozmernej adaptácie. Hlavnými závermi sú:

- **Kombinácia výhod:** Spojením charakteristík *XGBoost*u a techniky *LoRA* sme dosiahli model, ktorý zachováva interpretovateľnosť rozhodovacích stromov a zároveň získava flexibilitu v adaptácii na nové dáta.
- **Vysoká prenosnosť:** *XGBLoRA* sa ukazuje byť prenositeľným riešením pre rôzne domény, kde je potrebná rýchla adaptácia modelu bez straty kvality, čo potvrdili testy na viacerých typoch datasetov.

- **Efektívne učenie:** Dynamické nastavenie parametrov metódy *LoRA* umožňuje modelu učiť sa efektívnejšie z dostupných údajov, čím sa zlepšuje celková účinnosť bez potreby výrazného zvýšenia výpočtových zdrojov.

Navrhnutý model *XGBLoRA* má široké možnosti využitia v reálnych aplikáciách, obzvlášť v tých oblastiach, kde je potrebné rýchlo reagovať na nové dáta. Medzi hlavné oblasti praktického využitia patria:

- **Finančný sektor:** Predikcia úverových rizík a cenových pohybov na trhoch, kde *XGBLoRA* vďaka svojej adaptabilite dokáže lepšie reagovať na meniace sa trhové podmienky.
- **Zdravotníctvo:** Analýza klinických údajov a prognóza diagnóz, kde *XGBLoRA* pomáha zlepšiť presnosť predikcií napriek nedostatku rozsiahlych trénovacích údajov.
- **Priemyselná analýza:** Prediktívna údržba a kontrola kvality v priemyselných procesoch, kde *XGBLoRA* dokáže efektívne vyhodnocovať veľké množstvo senzorových údajov v reálnom čase.
- **Reklamné technológie:** Personalizácia a optimalizácia reklamných kampaní, kde *XGBLoRA* rýchlo aktualizuje predikcie na základe nových trendov a správania používateľov.

Pre ďalší vývoj a zlepšenie metódy *XGBLoRA* odporúčame venovať pozornosť nasledujúcim smerom:

1. Rozšírenie experimentov: Preskúmanie výkonnosti *XGBLoRA* na väčšom počte rôznorodých datasetov a porovnanie s ďalšími modernými prístupmi, čo pomôže overiť generalizovateľnosť navrhnutého modelu.
2. Optimalizácia hyperparametrov: Detailnejšia analýza nastavenia parametrov *LoRA* modulu (napríklad rozmerov nízkorozmernej matice), aby sa dosiahla ďalšia redukcia výpočtových nárokov bez straty presnosti modelu.

Literatúra

[1] Edward Hu et al. Lora: Low-rank adaptation of large language models. https://arxiv.org/pdf/2106.09685, 2021. Online.

[2] Yifei Zhang et al. Less is more: Extreme gradient boost rank-1 adaption for efficient finetuning of llms. https://arxiv.org/pdf/2410.19694, 2024. Online.

[3] Qingru Zhang et al. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. https://arxiv.org/pdf/2303.10512, 2023. Online.

[4] Aaron Xuxiang Tian et al. Fanlora: Fantastic loras and where to find them in large language model fine-tuning. https://aclanthology.org/2024.emnlp-industry.38.pdf, 2024. Online.

[5] Prakash Aryan. Resource-aware arabic llm creation: Model adaptation, integration, and multi-domain testing. https://arxiv.org/pdf/2412.17548, 2024. Online.

[6] Hasna Chouikhi et al. Gemmar: Enhancing llms through arabic instruction-tuning. https://arxiv.org/pdf/2407.02147, 2024. Online.

[7] Fan Jiang et al. Franken-adapter: Cross-lingual adaptation of llms by embedding surgery. https://arxiv.org/pdf/2502.08037, 2025. Online.

[8] Reece Shuttleworth et al. Lora vs full fine-tuning: An illusion of equivalence. https://arxiv.org/abs/2410.21228, 2024. Online.

[9] Aseer Ahmad Ansari. The lora family - rosa, dora, ada-lora, delta-lora, ... https://medium.com/@aseer-ansari/the-lora-family-rosa-dora-adalora-delta-lora-lora-vera-lorafa-lora-drop-7998 2024. Online.

[10] Menglin Yang et al. Low-rank adaptation for foundation models: A comprehensive review. https://arxiv.org/pdf/2501.00365, 2025. Online.

[11] Kerim Büyükakyüz et al. Olora: Orthonormal low-rank adaptation of large language models. https://arxiv.org/pdf/2406.01775, 2024. Online.

[12] Hongyun Zhou et al. Lora-drop: Efficient lora parameter pruning based on output evaluation. https://arxiv.org/pdf/2402.07721, 2024. Online.

[13] Yiran Zhao et al. Cross-lingual transfer with large language models via adaptive adapter merging. https://openreview.net/forum?id=y3CsNQal2l, 2024. OpenReview (ICLR 2024).

[14] Shaltiel Shmidman et al. Adapting llms to hebrew: Unveiling dictalm 2.0 with enhanced vocabulary and instruction capabilities. https://arxiv.org/pdf/2407.07080, 2024. Online.

[15] Klaudia Bałazy et al. Lora-xs: Low-rank adaptation with extremely small number of parameters. https://arxiv.org/pdf/2405.17604, 2024. Online.

Kód je zverejnený na GitHube: https://github.com/51nn3r/bp