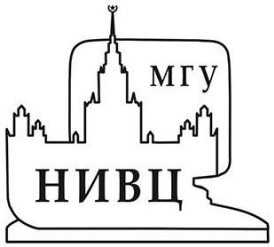


Fine-tuning, LoRa

к.ф.-м.н. Тихомиров М.М.

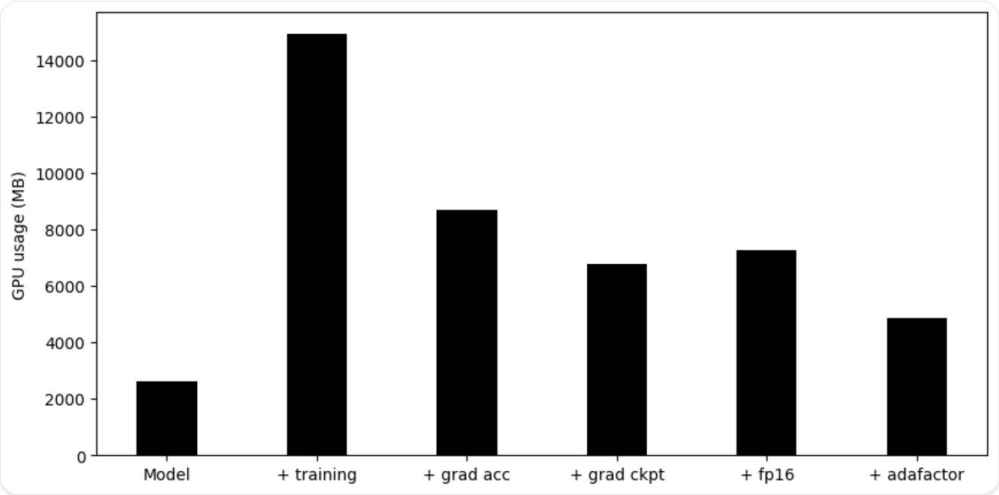
НИВЦ МГУ имени М. В. Ломоносова

Дообучение

- Иногда zero-shot и few-shot не достаточно.
- Модель **7b** параметров при загрузке на видеокарту в **fp16** занимает около **15gb** (и это даже не во время генерации или обучения), но хочется **fine-tuning**?
  - Полный fine-tuning, используя классические методы.
  - P-tuning, Prompt-tuning, Prefix-tuning.
  - LoRa / QLoRa.

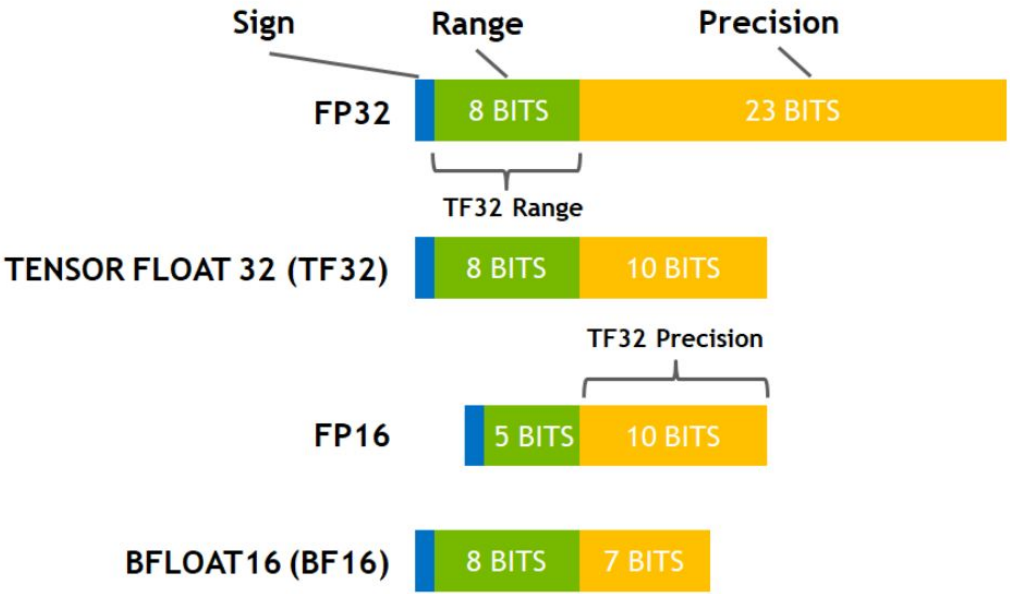
Fine-tuning: способы оптимизации

- Gradient Accumulation.
- Gradient Checkpointing.
  - Активации с forward pass можно не сохранять, а пересчитывать во время backward.
- Mixed-precision training (fp16 / bf16).
  - Выигрыш обычно сильно больше.
- Adafactor как альтернатива AdamW.
  - Могут быть проблемы со стабильностью



https://huggingface.co/docs/transformers/v4.18.0/en/performance

Различные типы чисел

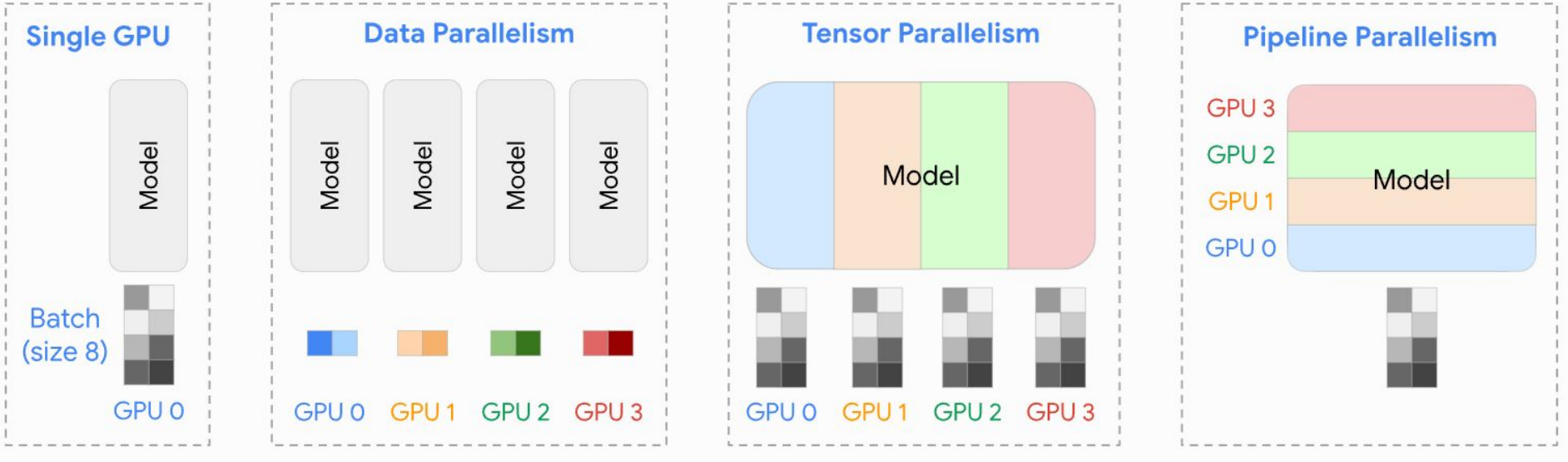


https://developer.nvidia.com/blog/accelerating-ai-training-with-tf32-tensor-cores/

Fine-tuning

- **Multi-gpu** система, V100 / **A100**,
- Использовать **DDP** (distributed data parallel), **gradient checkpointing**, **маленький batch size** + **accumulation gradients**,
- Пакет **DeepSpeed** от Microsoft:
  - Zero Stage 1: состояние оптимизатора распределено на разные GPU,
  - Zero Stage 2: + градиенты распределены на разные GPU,
  - Zero Stage 3: + веса модели распределены на разные GPU,
  - Параметр cpu offloading – выгружает часть данных с GPU на CPU в процессе обучения

Виды параллелизма

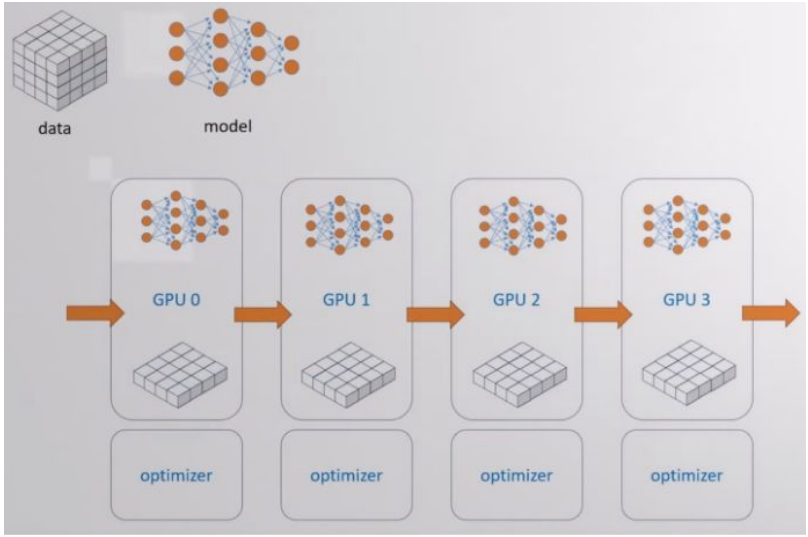


https://uvadlc-notebooks.readthedocs.io/en/latest/tutorial_notebooks/scaling/JAX/pipeline_parallel_simple.html

Data Parallelism

Data Parallelism (DP)- На каждой GPU копия модели, но данные из батча разделяются по разным GPU. Каждая GPU накапливает свои градиенты за шаг оптимизации, потом шаг синхронизации.

- Предназначен для 1 ноды
- Используется многопоточность, а не многопроцессорность
- Шаг оптимизации происходит на GPU 0, затем синхронизация на остальных

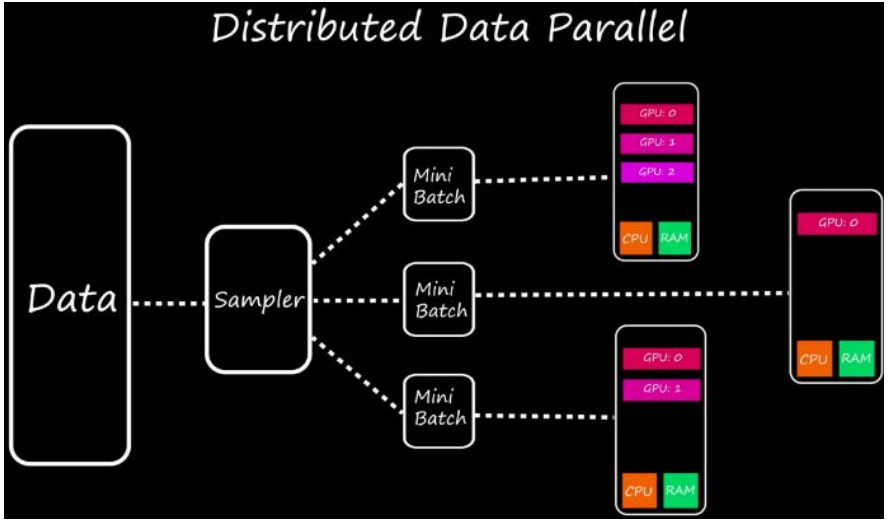


https://teletype.in/@kittybytes/rfMgEGnBsk1

Distributed Data Parallelism (DDP)

Логика полностью повторяет DP, но:

- Можно multi-node
- Используется multiprocessing
- Обычно используется DDP



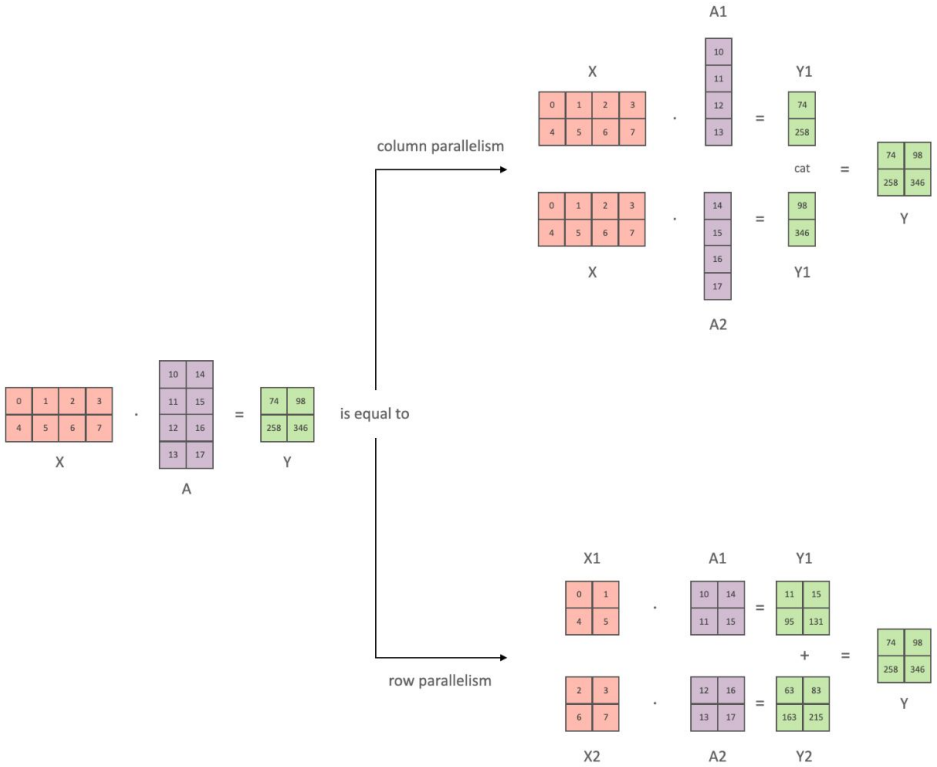
Когда можно и нужно использовать DDP:

- Модель влезает на GPU полностью + хватает места для процесса обучения

Tensor Parallelism

Tensor Parallelism - Модель “разрезается” на разные GPU через все слои. Каждая GPU считает свою часть тензорных операций, затем происходит агрегация.

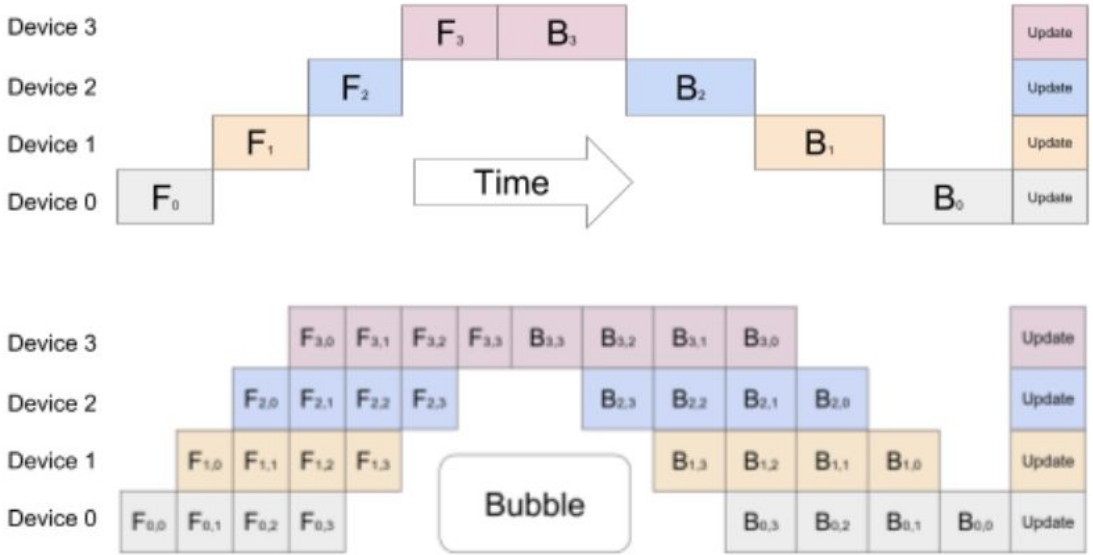
Используется для multigpu inference



https://huggingface.co/docs/transformers/v4.15.0/parallelism#tensor-parallelism

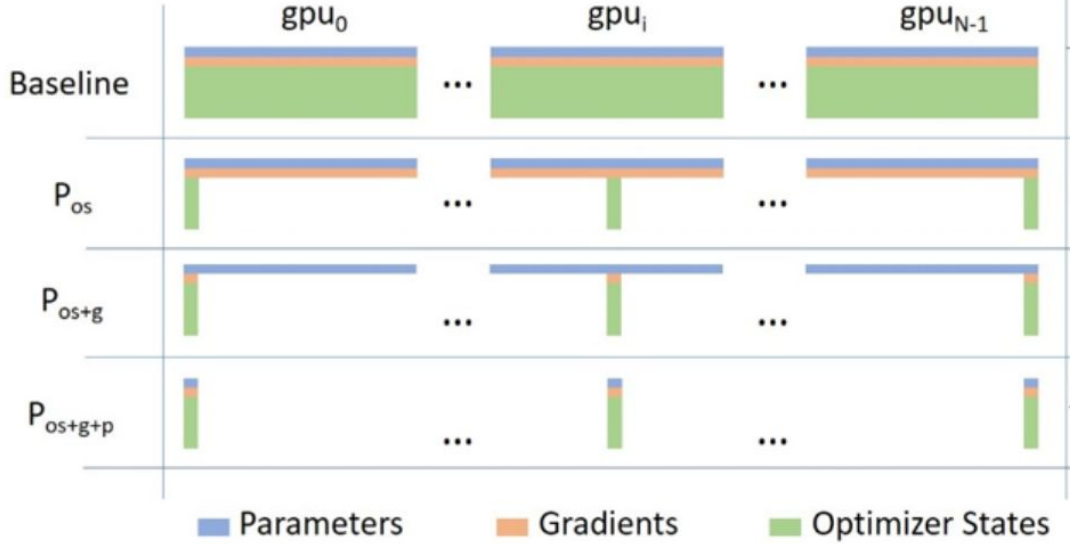
Pipeline Parallelism

Pipeline Parallelism - Модель разрезается по слоям на разные GPU. Конвейерная обработка.



https://blog.research.google/2019/03/introducing-gpipe-open-source-library.html

Deepspeed

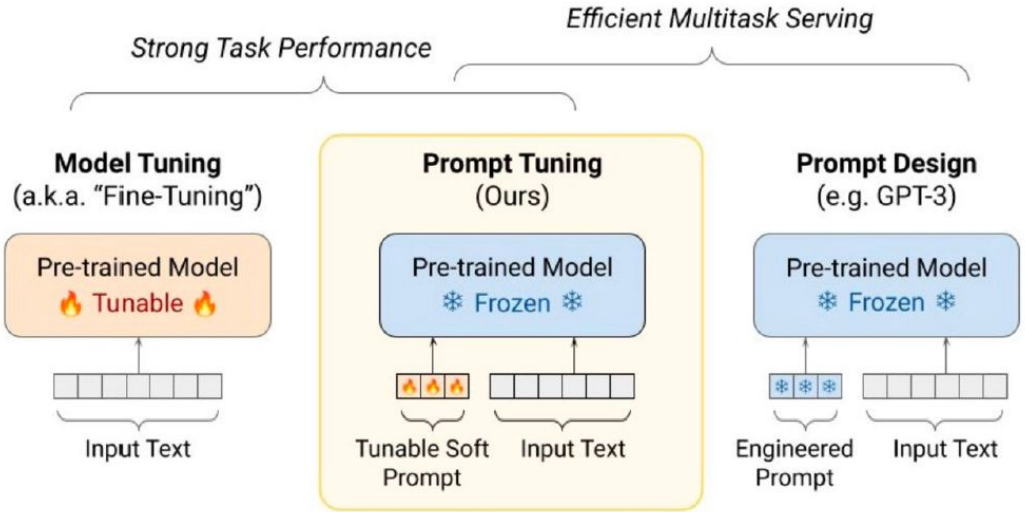


	Memory Consumption		Comm Volume
	Formulation	Specific Example K=12 Ψ=7.5B N_d=64	
Baseline	$(2+2+K)*\Psi$	120GB	1x
P_{os}	$2\Psi+2\Psi+\frac{K*\Psi}{N_d}$	31.4GB	1x
P_{os+g}	$2\Psi+\frac{(2+K)*\Psi}{N_d}$	16.6GB	1x
P_{os+g+p}	$\frac{(2+2+K)*\Psi}{N_d}$	1.9GB	1.5x

https://www.microsoft.com/en-us/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

Prompt-tuning

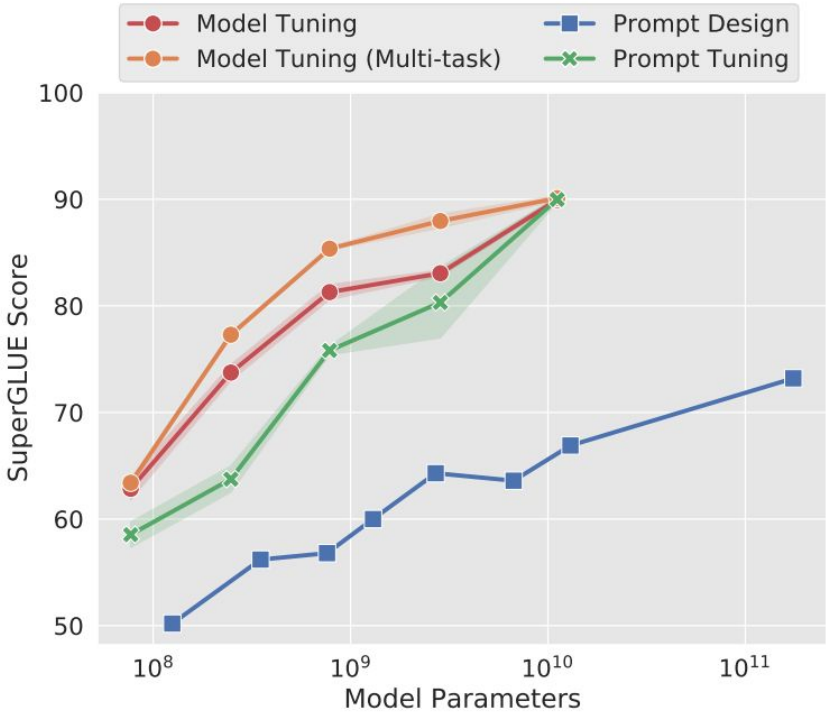
- Альтернатива дообучению,
- Вместо подбора слов (токенов) промпта, **подбираются входные эмбединги** для нескольких токенов входного текста.
- Напрямую обучаются “soft prompt” токены.



Lester B., Al-Rfou R., Constant N. The power of scale for parameter-efficient prompt tuning //arXiv preprint arXiv:2104.08691. – 2021.

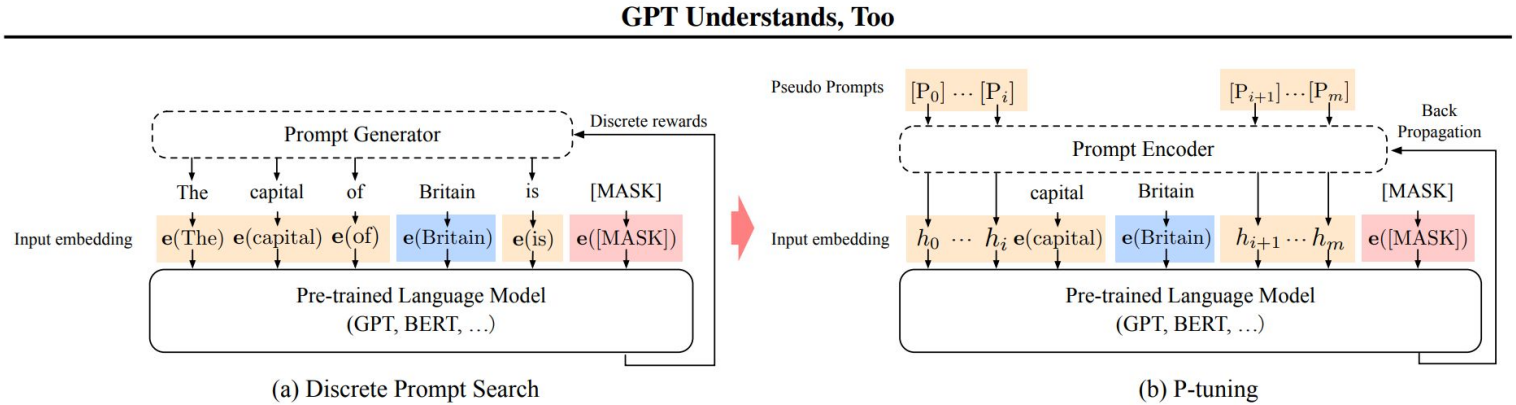
Prompt-tuning: результаты

Dataset	Domain	Model	Prompt	Δ
SQuAD	Wiki	94.9 ±0.2	94.8 ±0.1	−0.1
TextbookQA	Book	54.3 ±3.7	**66.8** ±2.9	+12.5
BioASQ	Bio	77.9 ±0.4	**79.1** ±0.3	+1.2
RACE	Exam	59.8 ±0.6	**60.7** ±0.5	+0.9
RE	Wiki	88.4 ±0.1	**88.8** ±0.2	+0.4
DuoRC	Movie	**68.9** ±0.7	67.7 ±1.1	−1.2
DROP	Wiki	**68.9** ±1.7	67.1 ±1.9	−1.8



P-tuning

- Еще одна альтернатива дообучению,
- Вместо подбора слов (токенов) промпта, **подбираются входные эмбединги** для нескольких токенов входного текста через специальный **prompt encoder**.



Промт энкодер нужно обучать!

Liu X. et al. GPT Understands, Too //arXiv preprint arXiv:2103.10385. – 2021.

P-tuning: результаты

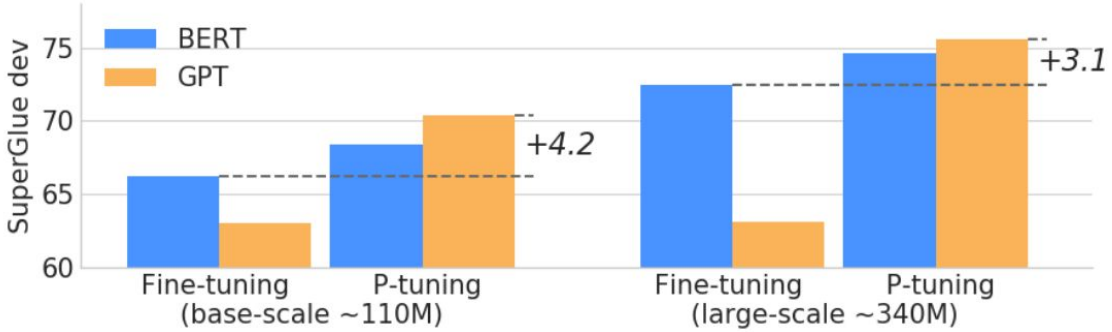
- Датасет LAMA-29
  - “пробинг” знаний моделей,

(Dante, born-in, Florence) ->
“Dante was born in [MASK].

Model	MP	FT	MP+FT	P-tuning
BERT-base (109M)	31.7	51.6	52.1	52.3 (+20.6)
-AutoPrompt (Shin et al., 2020)	-	-	-	45.2
BERT-large (335M)	33.5	54.0	55.0	54.6 (+21.1)
RoBERTa-base (125M)	18.4	49.2	50.0	49.3 (+30.9)
-AutoPrompt (Shin et al., 2020)	-	-	-	40.0
RoBERTa-large (355M)	22.1	52.3	52.4	53.5 (+31.4)
GPT2-medium (345M)	20.3	41.9	38.2	46.5 (+26.2)
GPT2-xl (1.5B)	22.8	44.9	46.5	54.4 (+31.6)
MegatronLM (11B)	23.1	OOM*	OOM*	**64.2** (+41.1)

* MegatronLM (11B) is too large for effective fine-tuning.

- Бенчмарк SuperGlue

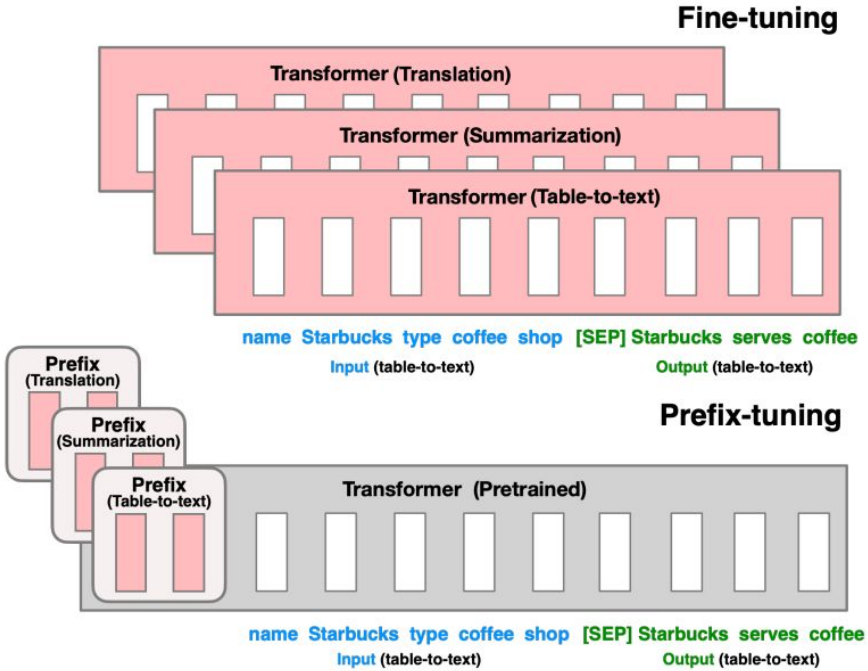


Prefix-tuning

- Продолжение предыдущих идей.
- Обучаются специальные эмбединги префикса, которые добавляются на каждом трансформер блоке.
- Эмбеддинги обучаются не напрямую, а через полносвязный слой.

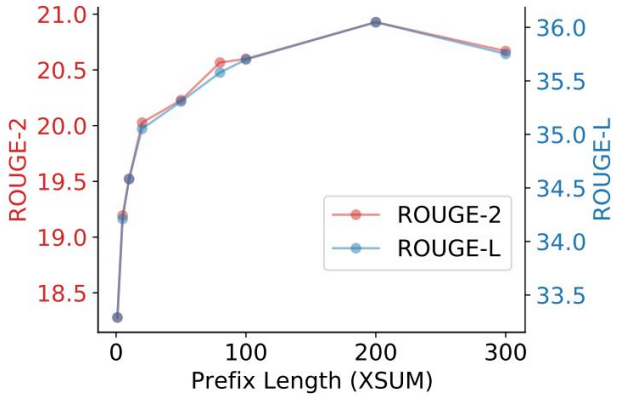
$$h_i = \begin{cases} P_\theta[i,:], & \text{if } i \in \mathsf{P}_{\text{idx}}, \\ \text{LM}_\phi(z_i, h_{<i}), & \text{otherwise.} \end{cases}$$

$$P_\theta[i,:] = \text{MLP}_\theta(P'_\theta[i,:]) \text{ by a smaller matrix } (P'_\theta)$$

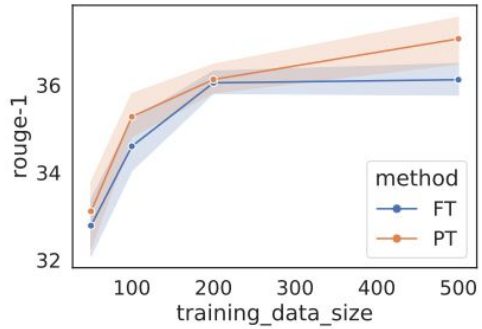


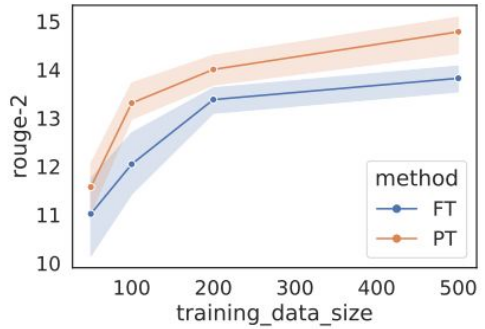
Li X. L., Liang P. Prefix-tuning: Optimizing continuous prompts for generation //arXiv preprint arXiv:2101.00190. – 2021.

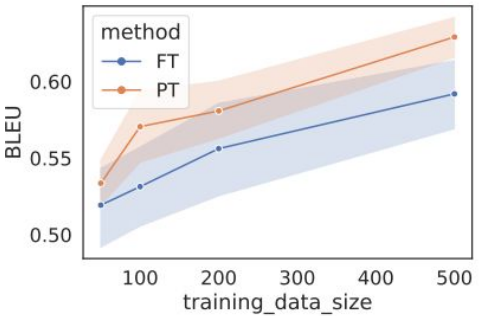
Prefix-tuning: результаты

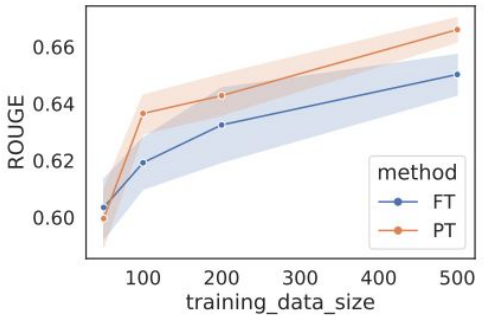


	R-1 ↑	R-2 ↑	R-L ↑
FINE-TUNE(Lewis et al., 2020)	45.14	22.27	37.25
PREFIX(2%)	43.80	20.93	36.05
PREFIX(0.1%)	42.92	20.03	35.05









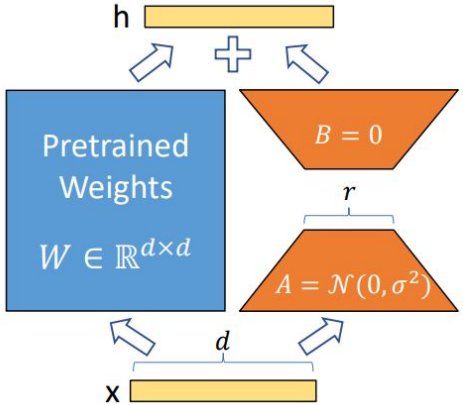
LoRa

- То, благодаря чему мы имеет saiga модели,
  - Первые версии **Saiga-7b** и **13b** обучались на всего одной **RTX 3090**,
- **Позволяет обучать** всю сеть, но при этом **уменьшая** количество обучаемых параметров **в 10,000 раз** (для GPT-3),
  - И требования к памяти GPU в 3 раза,
- Основная идея в том, чтобы не обучать все параметры модели, а только некоторую “добавку”, причем в **low-rank**,
- После слияния с моделью имеем новую модель **без дополнительных затрат** на работу.

Hu E. J. et al. Lora: Low-rank adaptation of large language models //arXiv preprint arXiv:2106.09685. – 2021.

LoRa: основная идея

$$W_0 + \Delta W = W_0 + BA, \text{ where } B \in \mathbb{R}^{d \times r}, A \in \mathbb{R}^{r \times k},$$



We then scale $\Delta W x$ by $\frac{\alpha}{r}$

- Обучаются матрицы B и A,
- Соответственно для каждого Q, K, V в attention можно применить подобный “трюк”.

LoRa: alpha

$$W_0 + \Delta W = W_0 + BA, \text{ where } B \in \mathbb{R}^{d \times r}, A \in \mathbb{R}^{r \times k}$$

$$h = W_0 x + \Delta W x = W_0 x + BAx$$

We then scale $\Delta W x$ by $\frac{\alpha}{r}$

LoRa: alpha

- Не сразу очевидный результат введения альфы - возможность изменить ее перед слиянием с моделью, в случае, если есть переобучение.
  - Добавку BA можно интерпретировать как накопленный градиент за обучение, меняя альфу мы меняем “learning rate” (хотя “шаг” один).

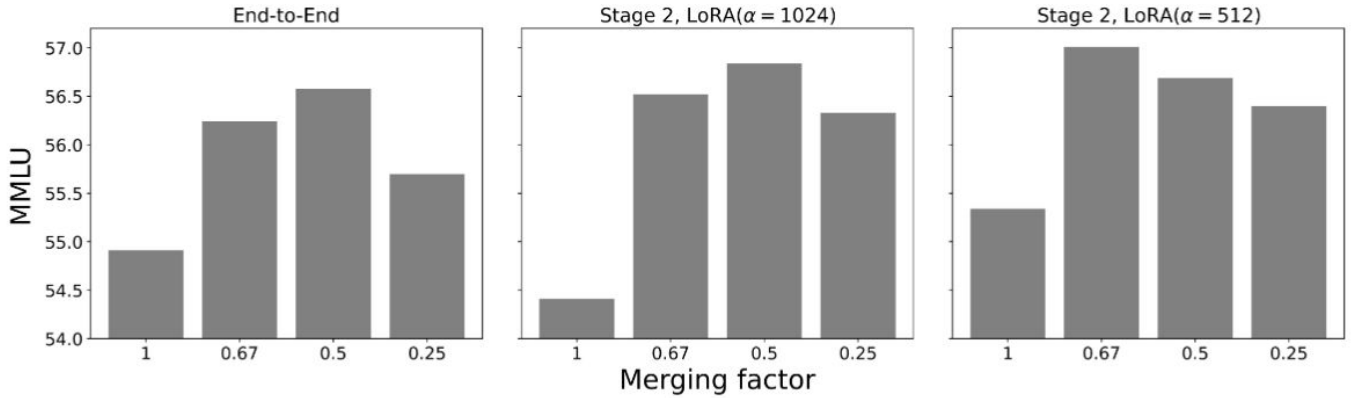


Figure 5. The effect of LoRA adapter merging factor on model performance

LoRa: результаты

Model&Method	# Trainable Parameters	WikiSQL Acc. (%)	MNLI-m Acc. (%)	SAMSum R1/R2/RL
GPT-3 (FT)	175,255.8M	**73.8**	89.5	52.0/28.0/44.5
GPT-3 (BitFit)	14.2M	71.3	91.0	51.3/27.4/43.5
GPT-3 (PreEmbed)	3.2M	63.1	88.6	48.3/24.2/40.5
GPT-3 (PreLayer)	20.2M	70.1	89.5	50.8/27.3/43.5
GPT-3 ($\text{Adapter}^{\text{H}}$)	7.1M	71.9	89.8	53.0/28.9/44.8
GPT-3 ($\text{Adapter}^{\text{H}}$)	40.1M	73.2	**91.5**	53.2/29.0/45.1
GPT-3 (LoRA)	4.7M	73.4	**91.7**	**53.8/29.8/45.9**
GPT-3 (LoRA)	37.7M	**74.0**	**91.6**	53.4/29.2/45.1

LoRa: какие матрицы дообучать

	# of Trainable Parameters = 18M						
Weight Type	W_q	W_k	W_v	W_o	W_q, W_k	W_q, W_v	W_q, W_k, W_v, W_o
Rank r	8	8	8	8	4	4	2
WikiSQL ($\pm 0.5\%$)	70.4	70.0	73.0	73.2	71.4	**73.7**	**73.7**
MultiNLI ($\pm 0.1\%$)	91.0	90.8	91.0	91.3	91.3	91.3	**91.7**

Unsloth

- Известный фреймворк для более быстрого instruction-tuning
- Обычный сценарий: LoRa + int4 + unsloth
- Плюсы: быстрый, минимальные изменения в коде
- Минусы: только single-gpu

1 A100 40GB	Dataset	Hugging Face	+ Flash Attention 2	Unsloth	VRAM reduction
Code Llama 34b	Slim Orca	1x	1.01x	**1.94x**	-22.7%
Llama-2 7b	Slim Orca	1x	0.96x	**1.87x**	-39.3%
Mistral 7b	Slim Orca	1x	1.17x	**1.88x**	-65.9%
Tiny Llama 1.1b	Alpaca	1x	1.55x	**2.74x**	-57.8%
DPO with Zephyr	Ultra Chat	1x	1.24x	**1.88x**	-11.6%

Free Colab T4	Dataset	Hugging Face	+ Pytorch 2.1.1	Unsloth	VRAM reduction
Llama-2 7b	OASST	1x	1.19x	**1.95x**	-43.3%
Mistral 7b	Alpaca	1x	1.07x	**1.56x**	-13.7%
Tiny Llama 1.1b	Alpaca	1x	2.06x	**3.87x**	-73.8%
DPO with Zephyr	Ultra Chat	1x	1.09x	**1.55x**	-18.6%

Паддинг

Важный аспект при подготовке данных для обучения.

- Алгоритмы требуют на вход batch размером (bs, seq_len), элементами которого являются токены.
- Разные последовательности имеют разную длину в токенах, но надо собрать тензор
- Как решение - добавление спец. токена pad
  - Маска внимания при этом модифицируется
  - Паддить можно как слева, так и справа.
    - При instruction-tuning обычно слева, так как batched inference возможен только с left padding
    - При pretraining pad вообще не используется в большинстве случаев

Домашнее задание 4

Цель задания:

Осуществить fine-tuning с использованием LoRa.

- Берете вашу “задачу” с прошлого ДЗ и разбиваете данные на train и test (если они предварительно не разбиты)
- За “базу” берете уже инструктивную модель, например, qwen-2.5-3B-instruct
- Обучаете модель на “разумном” количестве примеров, чтобы не было слишком долго
- Сравниваете качество на test части до и после.

Задание: оценивание и сроки

- Срок 1 неделя: до 10 ноября 23:59.
- Присылать на tikhomirov.mm@gmail.com
  - Название письма: Practical LLM: Задание 4, fine-tuning.
  - В письме Ваше полное ФИО, группа, решение и краткий отчет по нему в PDF.
- Оценка по шкале “-/-+/+-/+/++”.
  - ++ за те решения, которые особо мне понравятся чем-либо.