Тензорные и Матричные разложения

Москворецкий Виктор

Про матрицы
Сингулярное разложение
Практические применения
Тензорные разложения

Нейросеть - набор матриц
Ранг матрицы

Определение SVD
Усеченное разложение
Оптимальный ранг
Примеры BERT и LLaMA-2

LadaBERT
Fisher-SVD
LASER
KroneckerBERT

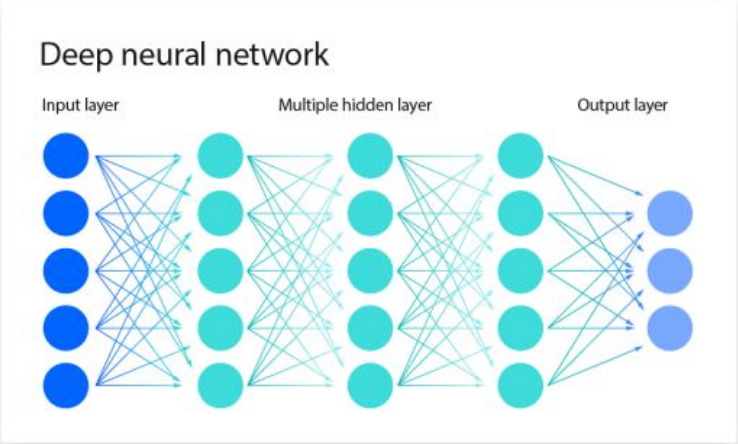
Определение тензора
CP разложение
Tucker Разложение
Tensor Train разложение

>>> Матрицы

Нейросеть - набор матриц

Все нейросети - “модные” матричные умножения

- Полносвязная сеть - Матричные умножения с нелинейностями между ними
- Сверточная сеть - Тензорные умножения с нелинейностями между ними
- Рекуррентная сеть - Рекуррентные Матричные умножения с нелинейностями между ними
- Механизм внимания - Матричные умножения с софтмаксом
- Современные трансформеры - Глубокая полносвязная сеть с механизмом внимания



Нейросеть - Последовательное преобразование признакового пространства
Цель: сделать сложное пространство приятным для решения задачи

$$\varphi_k(W_k \cdot \ldots \cdot \varphi_2(W_2 \cdot \varphi_1(W_1 \cdot x)))$$

Ранг матрицы

Линейный слой принимающий n признаков и выдающий m признаков → $W^{n \times m}$

Ранг - это настоящая размерность выходного пространства → $W^{n \times m}$

Пример:

$$W_1, \quad rank(W_1) = 1, \quad W_1 \in R^{2 \times 2} \qquad \begin{pmatrix} 1 & 2 \\ 0 & 0 \end{pmatrix}$$

$$(x, y) \xrightarrow{W} (x, 2x)$$

Вывод: Может сократить размер матрицы без потери информации

Эквивалентные определения ранга

система векторов называется линейно независимой, когда $\alpha_1 v_1 + \cdots + \alpha_n v_n = 0$ имеет только тривиальное решение, то есть все $\alpha_i = 0$

Столбцовый: максимальное количество линейно независимых столбцов

Строковый: максимальное количество линейно независимых строк

Факториальный: Минимальная размерность в разложении

$$\min\{k | W = BC, \text{ where } B \in \mathbb{R}^{n \times k}, C \in \mathbb{R}^{k \times m}\}$$

Тензорный: Минимальное количество тощих матриц необходимых для разложения

$$\min\{k | W = x_1 y_1^T + \ldots + x_k y_k^T, x_i \in \mathbb{R}^n, y_i \in \mathbb{R}^m\}$$

Повышаем эффективность

- Разложения могут сократить количество параметров. Хранение происходит эффективнее
- При определенных условиях можно ускорить инференс. Если суммарное число операций станет меньше
- Может служить регуляризацией, потенциально повышая качество. А возможно и динамику обучения
- Можем сохранить всю основную информацию в наших матрицах

>>> Сингулярное разложение

Сингулярное разложение

- Основной вид разложения
- Всегда существует
- Хорошо численно считается
- Имеет приятную интерпретацию
- Широко используется
- Определен для любых матриц

$$W = U\Sigma V^*$$

$$UU^* = I \qquad VV^* = I$$

- U и V - унитарные. Левые и правые сингулярные вектора
- Σ - диагональная с сингулярными значениями

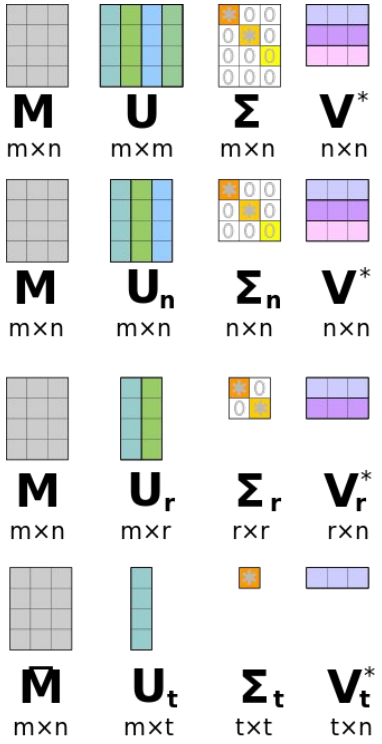
Низкоранговая аппроксимация

$$\text{minimize} \quad \text{over } \widehat{D} \quad \|D - \widehat{D}\|_{\mathrm{F}} \quad \text{subject to} \quad \text{rank}\left(\widehat{D}\right) \leq r$$

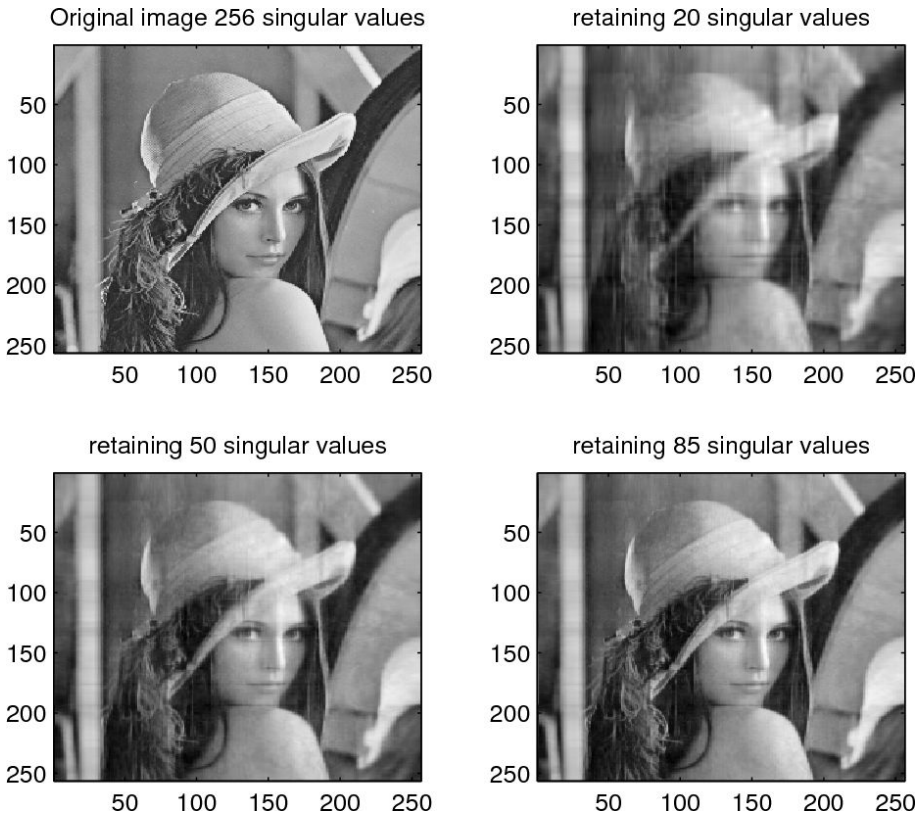
Задача: Найти низкоранговую матрицу, чтобы она сохраняла информацию

Решение: Усеченное сингулярное разложение $\hat{D} = U_r \Sigma_r V_r^*$

- Делаем SVD
- Берем первые r компонент
- Лучшее решение данной задачи
- Аналитически доказанное



Усеченное сингулярное разложение



Оптимальный ранг разложения

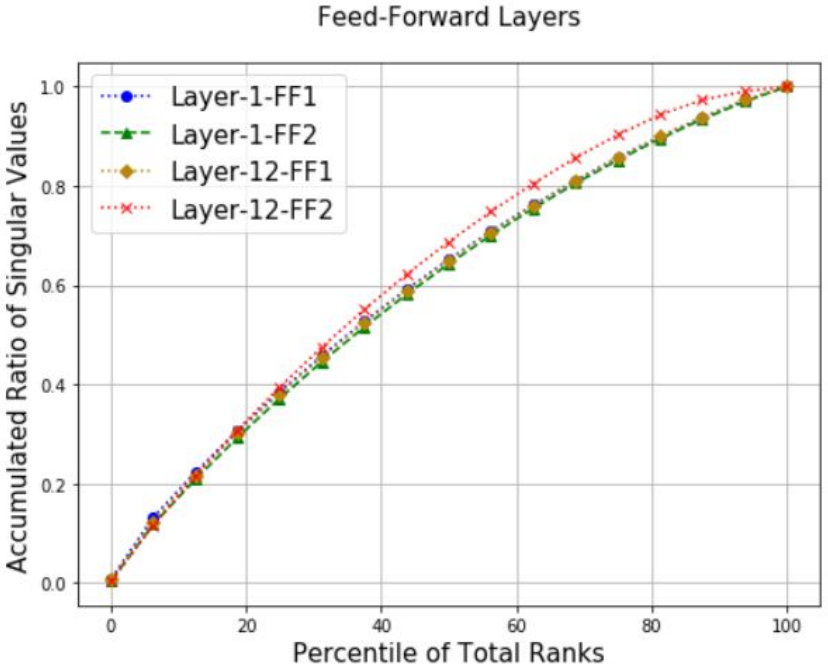
- Изначальная память: $N \times M$
- Размерность сингулярного разложения
  $U_r \in R^{N \times r} \quad \sum_r \in R^r \quad V_r^* \in R^{r \times M}$
- Память после сингулярного разложения
  $(N \times r) + r + (r \times M) = r(N + M + 1)$
- Условие оптимальности
  $r(N + M + 1) < NM$
- Необходимый ранг

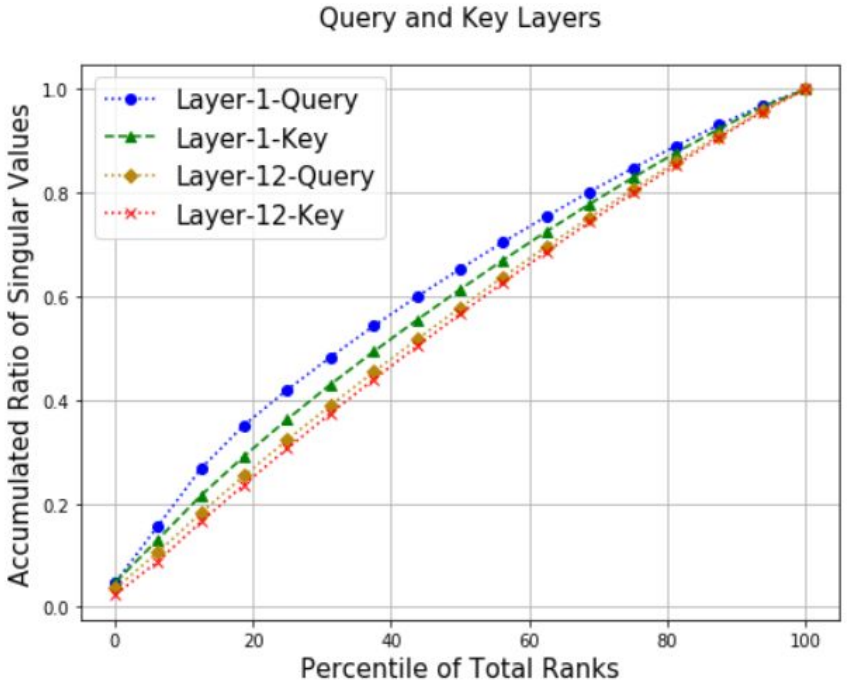
$$r < \frac{NM}{N + M + 1}$$

BERT base SA, N=M=768 -> r < 383

Llama2-7b SA, N=M=4096 -> r < 2048

Усеченное сингулярное разложение (BERT)



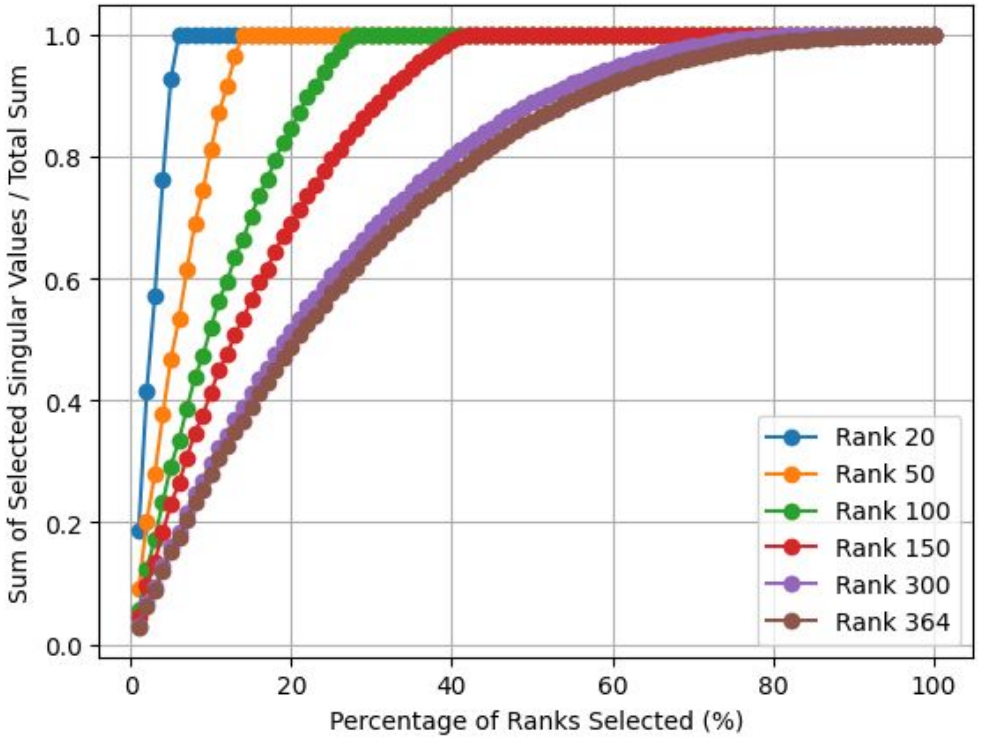


Какую из матриц лучше сжимать?

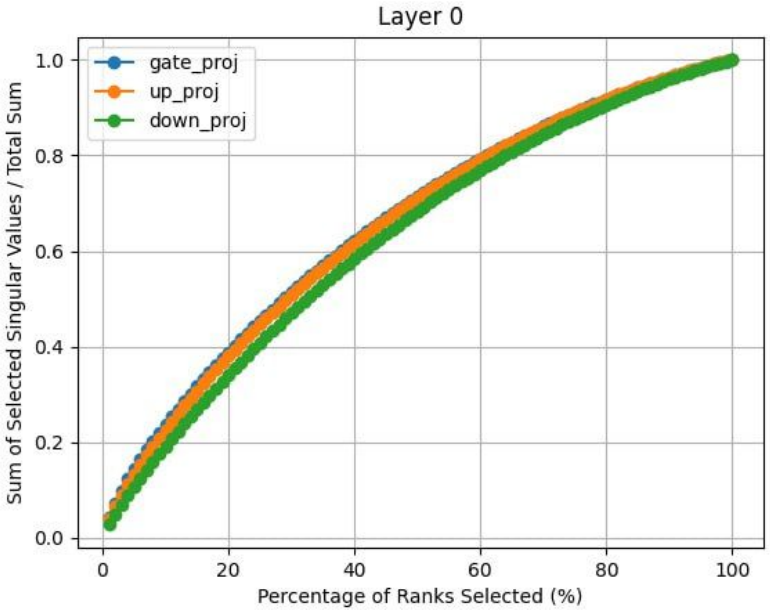
Ответ: Layer 12 FF2

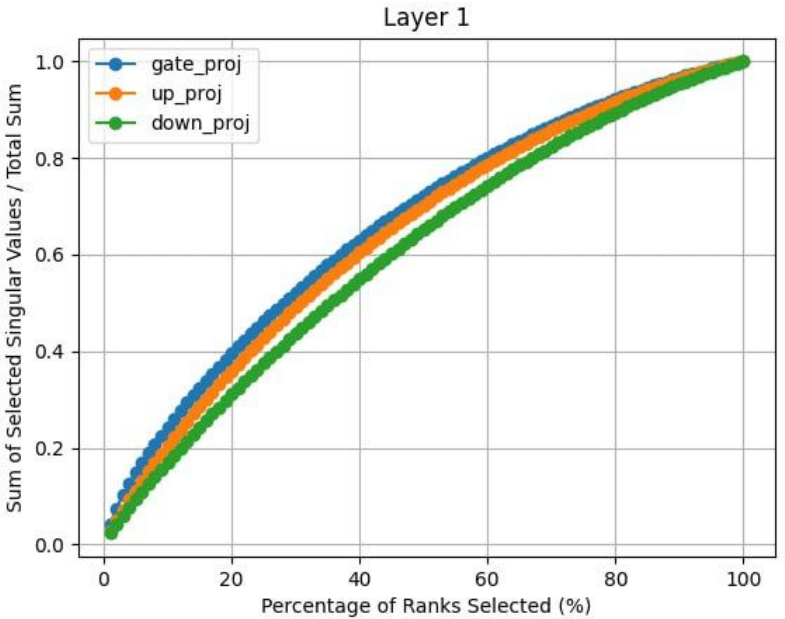
Усеченное сингулярное разложение (BERT)

Как выглядит матрица, которую удобно сжимать?



Усеченное сингулярное разложение (LLaMA2)

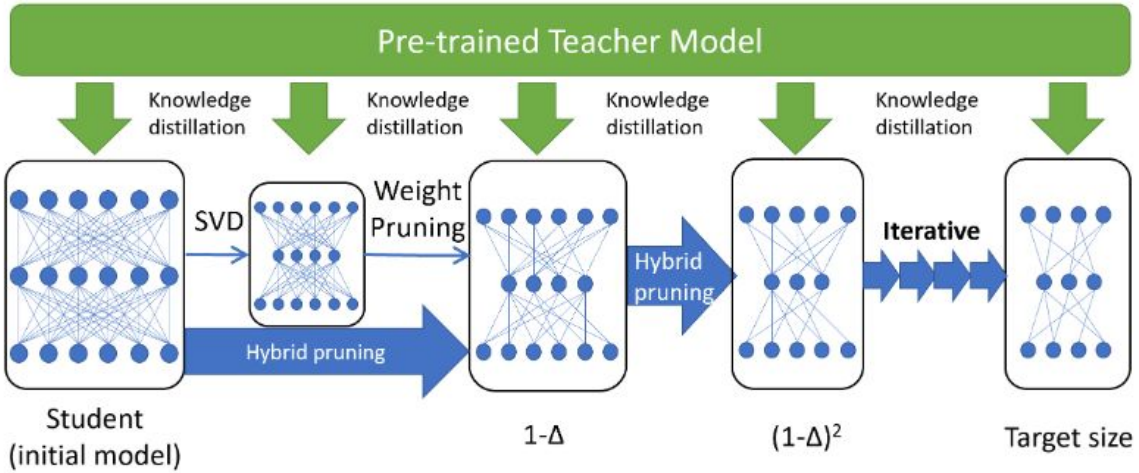




>>> Практические применения

LADA BERT

- Итеративный Подход
- Сингулярное разложение
- Прунинг
- Дообучение
- Дистилляция



Источник: Mao, Yihuan, et al. "Ladabert: Lightweight adaptation of bert through hybrid model compression." arXiv preprint arXiv:2004.04124 (2020).

LADA BERT

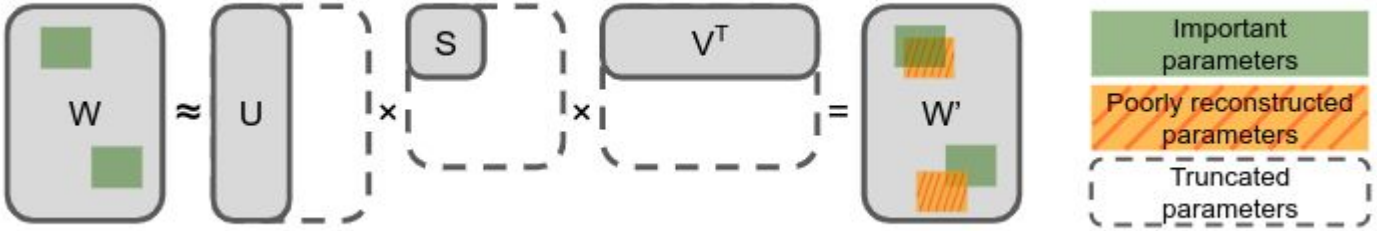
- Основной прирост за счет дистилляции
- Отдельно прунинг и разложение портят модель

Table 3: Performance comparison on various model sizes

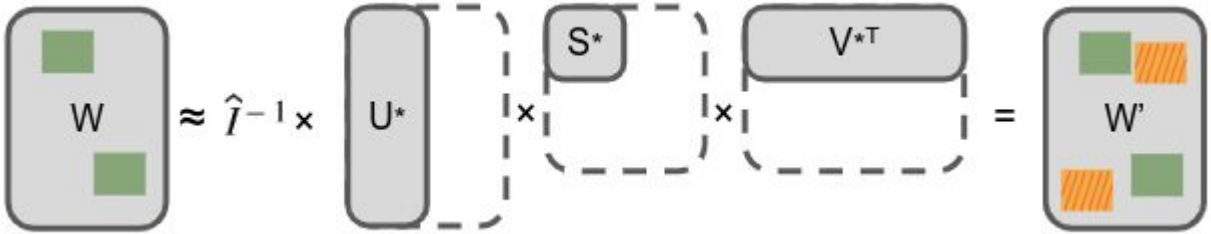
Algorithm	MNLI-m	MNLI-mm	SST-2	QQP	QNLI	#Params	Ratio
BERT-Base	**84.6**	**83.4**	**93.5**	**71.2/-**	**90.5**	110M	×1.0
LadaBERT-1	**83.5**	**82.5**	**92.8**	**70.7/88.9**	**89.6**	44M	×2.5
BERT-FT	74.8	74.3	86.4	65.8/86.9	84.3	44M	×2.5
BERT-KD	75.4	74.8	86.9	67.3/87.6	84.0	44M	×2.5
BERT-PKD	76.7	76.3	87.5	68.1/87.8	84.7	44M	×2.5
Weight pruning	82.8	81.6	92.3	70.1/88.5	88.9	44M	×2.5
matrix factorization	77.7	77.4	87.6	65.7/87.2	84.3	44M	×2.5
Hybrid pruning	81.2	80.0	90.0	68.0/87.5	83.3	44M	×2.5
LadaBERT-2	**83.1**	**82.2**	**91.8**	**69.9/87.9**	**88.2**	22M	×5.0
Weight pruning	75.9	75.6	84.8	60.3/83.5	81.7	22M	×5.0
matrix factorization	71.8	71.8	82.8	60.3/83.5	75.4	22M	×5.0
Hybrid pruning	76.1	75.3	85.4	64.9/85.8	80.6	22M	×5.0
LadaBERT-3	**82.1**	**81.8**	**89.9**	**69.4/87.8**	84.5	15M	×7.5
TinyBERT	80.9	79.5	89.5	65.4/87.5	77.9	15M	×7.5
BERT-Small	75.4	74.9	87.6	66.5/-	**84.8**	15M	×7.5
Weight pruning	69.1	68.8	81.8	59.7/82.9	76.4	15M	×7.5
matrix factorization	60.2	60.0	81.3	58.5/82.0	62.2	15M	×7.5
Hybrid pruning	71.9	71.0	83.5	62.3/84.7	73.8	15M	×7.5
LadaBERT-4	**75.8**	**76.1**	84.0	67.4/86.6	**75.1**	11M	×10.0
Distilled-BiLSTM	73.0	72.6	**90.7**	**68.2/88.1**	-	10M	×10.8
Weight pruning	64.9	65.1	80.4	56.9/80.5	62.7	11M	×10.0
matrix factorization	59.9	59.6	79.2	57.8/81.9	62.2	11M	×10.0
Hybrid pruning	68.4	67.9	81.5	58.6/83.5	63.2	11M	×10.0

Fisher Weighted SVD

- Сингулярное разложение не знает что важно модели



- Покажем ему что важно $\min_{A,B} ||\hat{I}W - \hat{I}AB||_2.$



Источник: Hsu, Yen-Chang, et al. "Language model compression with weighted low-rank factorization." arXiv preprint arXiv:2207.00112 (2022).

Fisher Weighted SVD

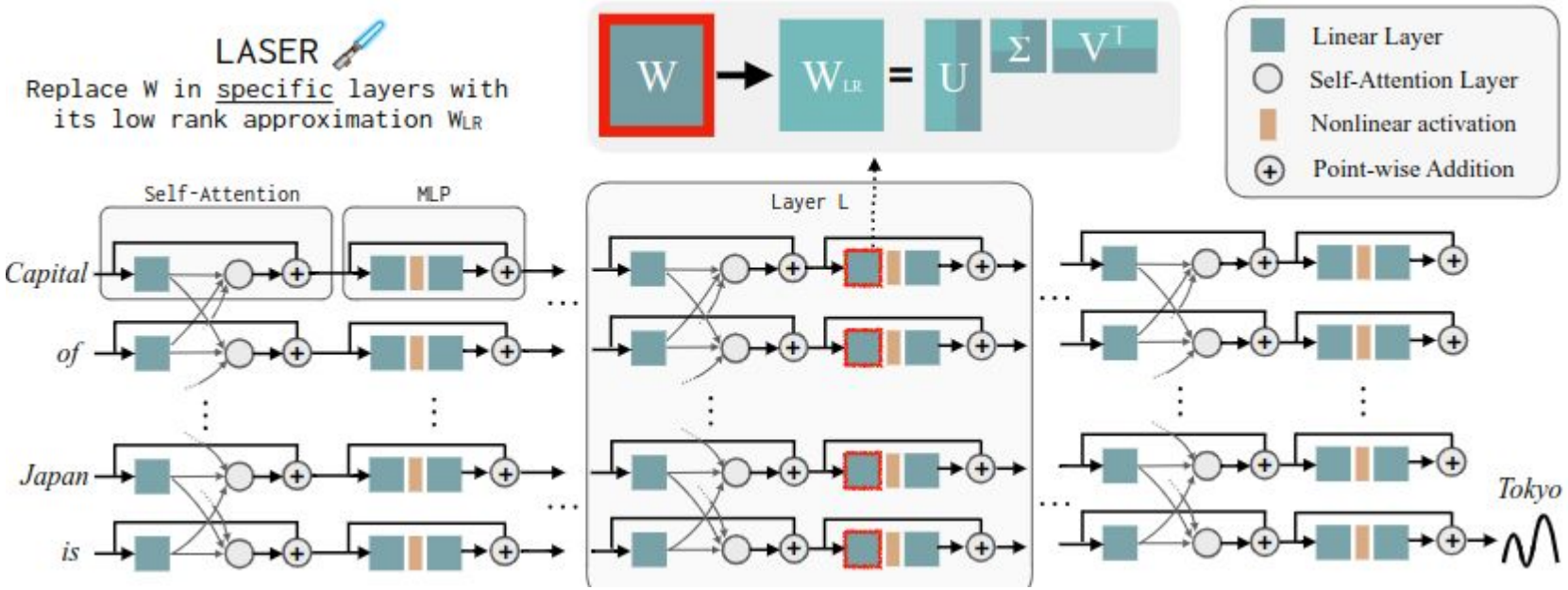
- Оцениваем информацию Фишера

$$I_w = E\left[\left(\frac{\partial}{\partial w}\log p(D|w)\right)^2\right] \approx \frac{1}{|D|}\sum_{i=1}^{|D|}\left(\frac{\partial}{\partial w}\mathcal{L}(d_i;w)\right)^2 = \hat{I}_w.$$

- Информация в строке = сумма по столбцам

	Model	#Param	CoNLL	CoLA	MNLI	MRPC	QNLI	QQP	SST-2	STS-B	G-Avg	A-Avg
Original	BERT_{base}	109.5M	94.1	56.2	84.7	87.4	91.3	87.8	93.0	88.5	84.1	85.4
Path-1	DistilBERT	67.0M	93.2	49.8	82.2	88.7	89.3	86.7	90.4	86.1	81.9	83.3
	MiniLMv2	67.0M	92.2	43.3	84.0	89.1	90.6	86.7	91.4	88.1	81.9	83.2
Path-2	BERT-PKD	67.0M	–	45.5	81.3	85.7	88.4	88.4	91.3	86.2	81.0	–
	BERT+SVD	66.5M	12.0	2.7	35.6	61.4	37.2	60.0	76.7	26.8	42.9	39.0
	+fine-tuning	66.5M	92.4	40.5	82.8	84.1	89.6	87.3	90.9	85.7	80.1	81.6
	BERT+FWSVD	66.5M	49.6	13.5	52.8	81.2	52.2	65.7	82.1	68.6	59.4	58.2
	+fine-tuning	66.5M	93.2	49.4	83.0	88.0	89.5	87.6	91.2	87.0	**82.2**	**83.6**

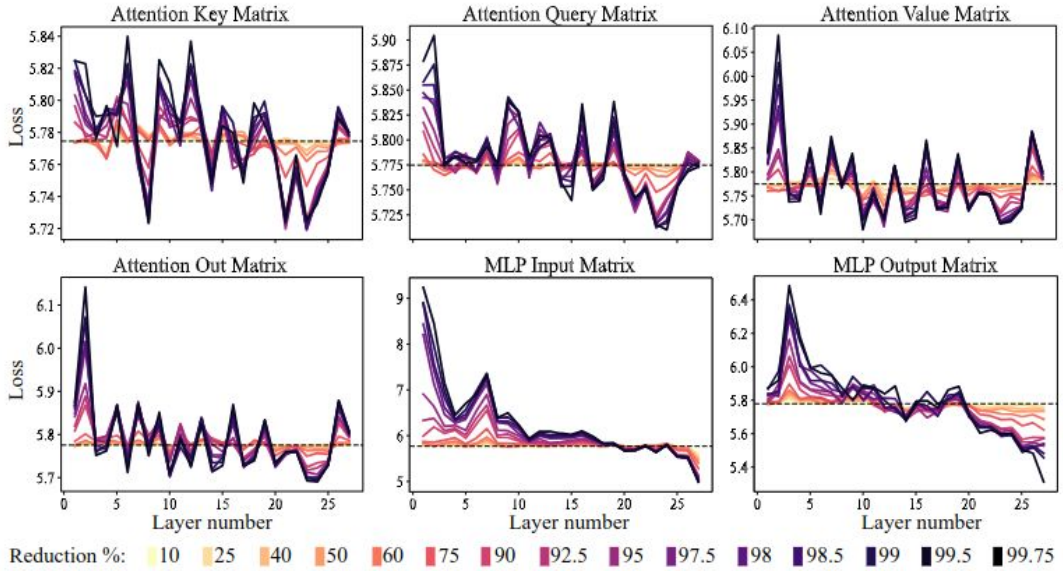
LASER: SVD to LLM



Источник: Sharma, Pratyusha, Jordan T. Ash, and Dipendra Misra. "The truth is in there: Improving reasoning in language models with layer-selective rank reduction." arXiv preprint arXiv:2312.13558 (2023).

LASER: SVD to LLM

- Разные слои ведут себя по разному
- Разные матрицы ведут себя по разному
- Некоторые очень чувствительные
- Некоторые совсем не чувствительные



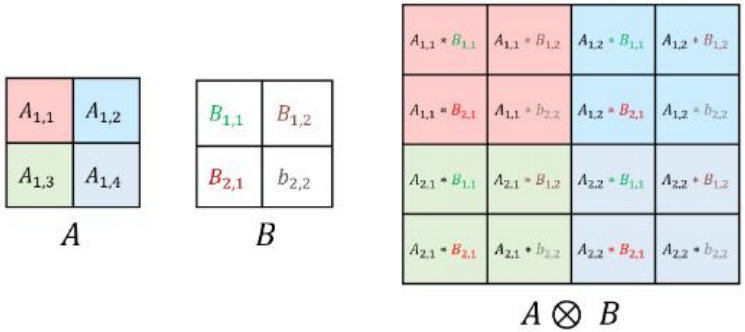
LASER: SVD to LLM

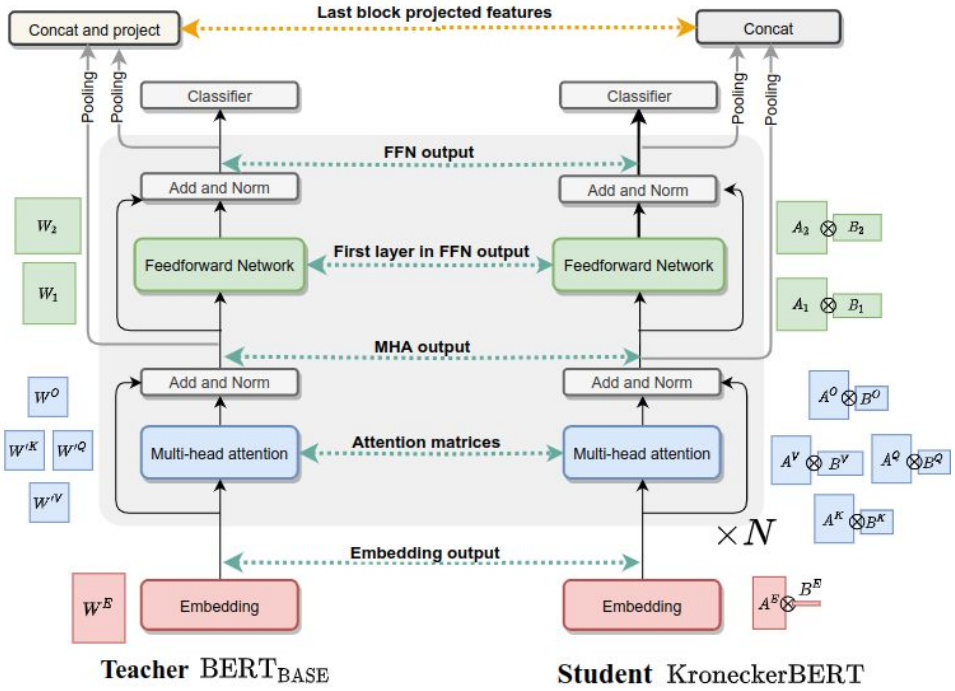
- Небольшое усечение
- Прирост к “честности”
- Работает на всех видах моделей

Dataset		Model Name					
		Roberta	LASER	GPT-J	LASER	LLama2	LASER
CounterFact	Acc	17.3	**19.3**	13.1	**24.0**	35.6	**37.6**
	Loss	5.78	**5.43**	5.78	**5.05**	3.61	**3.49**
HotPotQA	Acc	6.1	**6.7**	**19.6**	19.5	16.5	**17.2**
	Loss	10.99	**10.53**	3.40	**3.39**	3.15	**2.97**
FEVER	Acc	50.0	**52.3**	50.2	**56.2**	59.3	**64.5**
	Loss	2.5	**1.76**	**1.24**	1.27	1.02	**0.91**
Bios Gender	Acc	87.5	**93.7**	70.9	**97.5**	75.5	**88.4**
	Loss	**0.87**	1.13	**3.86**	4.20	3.48	**2.93**
Bios Profession	Acc	64.5	**72.5**	75.6	**82.1**	85.0	**86.7**
	Loss	**4.91**	6.44	**4.64**	4.91	4.19	**4.05**
TruthfulQA	Acc	56.2	56.2	54.9	**55.6**	50.5	**56.2**
	Loss	1.60	**1.42**	1.02	**1.01**	**0.95**	1.04
BigBench-Epistemic Reasoning	Acc	37.1	**41.8**	37.1	**38.3**	44.8	**63.4**
	Loss	9.39	**6.80**	0.74	**0.62**	0.78	**0.73**
BigBench-WikidataQA	Acc	28.0	**30.7**	51.8	**65.9**	59.5	**62.0**
	Loss	9.07	**7.69**	3.52	**2.86**	2.40	**2.31**

KroneckerBert

- Основано на Кронекеровом произведении
- Может эффективно раскладывать до более низкого ранга
- Применяем к каждой матрице





Источник: Tahaei, Marzieh S., et al. "Kroneckerbert: Learning kronecker decomposition for pre-trained language models via knowledge distillation." arXiv preprint arXiv:2109.06243 (2021).

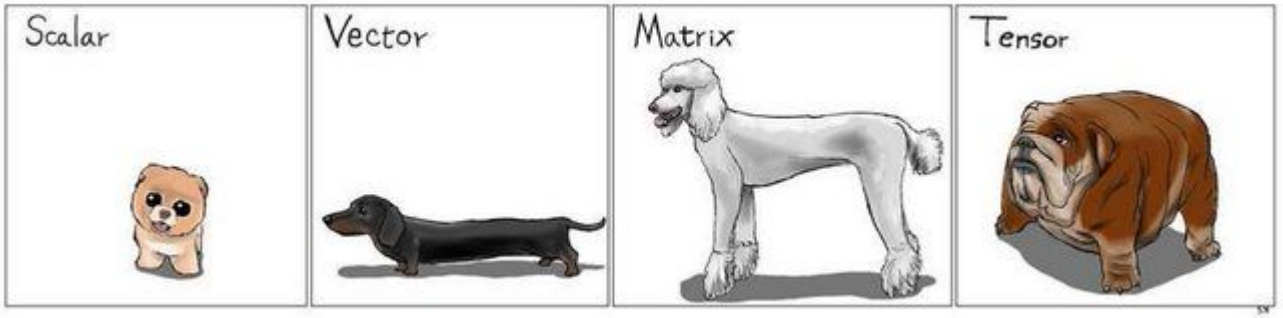
KroneckerBert

- Может работать лучше чем LADA BERT
- Сжатие в 19 раз
- Основной прирост за счет дистилляции

Pre-training	Fine-tuning	MNLI-m	SST-2	MRPC
None	No KD	66.0	81.3	68.3
None	KD	80.7	86.2	70.3
KD	No KD	77.0	87.2	78.17
KD	KD	**82.8**	**90.6**	**86.6**

Model	Params	MNLI-(m/mm)	SST-2	MRPC	CoLA	QQP	QNLI	RTE	STS-B	Avg
$BERT_{BASE}$	108.5M	83.9/83.4	93.4	87.9	52.8	71.1	90.9	67	85.2	79.5
$BERT_4$-PKD	52.2M	79.9/79.3	89.4	82.6	24.8	70.2	85.1	62.3	79.8	72.6
TinyBERT	14.5M	**82.5/81.8**	**92.6**	86.4	**44.1**	**71.3**	87.7	66.6	80.4	77.0
$LadaBERT_3$	15M	82.1/81.8	89.9	-	-	69.4	84.5	-	-	-
$KroneckerBERT_8$	14.3M	**82.9**/81.7	91.2	**88.5**	31.2	70.8	**88.4**	**66.9**	**83.1**	76.1
SharedProject	5.6M	76.4/75.2	84.7	84.9	-	-	-	-	-	-
$LadaBERT_4$	11M	75.8/76.1	84.0	-	-	67.4	75.1	-	-	-
$KroneckerBERT_{19}$	5.7M	**79.4/81.6**	**89.2**	**86.9**	25.8	**69.2**	**86.2**	62.7	78.2	73.1

>>> Тензоры



Что такое тензор

- Отображение из одного пространства в другое
- Или просто набор матриц

- Более общий вид
- Может встречаться в DL приложениях

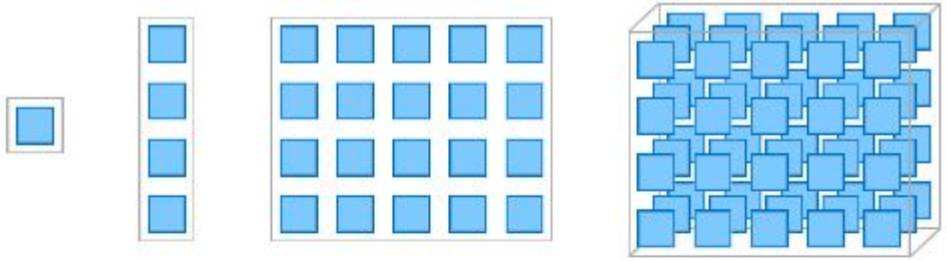


Figure 3: $x \in \mathbb{R}$, $\boldsymbol{x} \in \mathbb{R}^4$, $\boldsymbol{X} \in \mathbb{R}^{4\times5}$, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{4\times5\times3}$

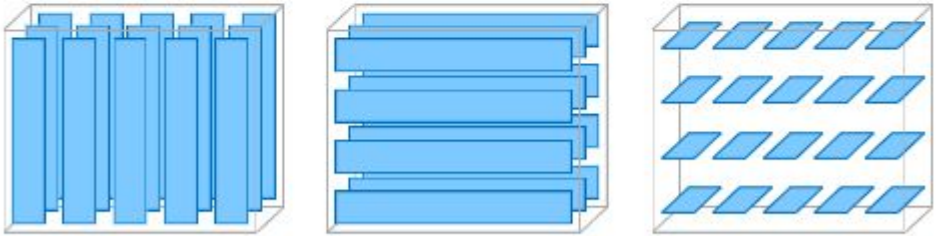


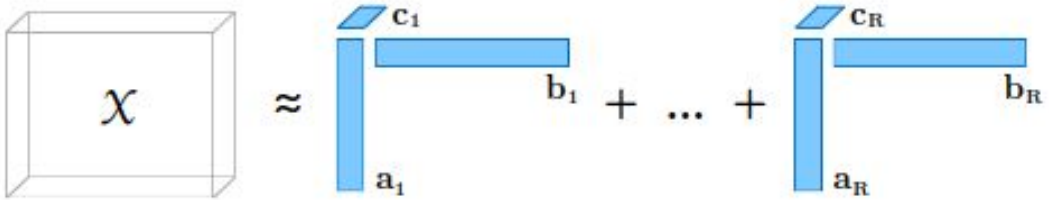
Figure 4: Column, row, and tube fibers of a mode-3 tensor

CP-разложение

- Сумма одноранговых тензоров
- Каждый член получен из тощих матриц
- Буквально определение ранга

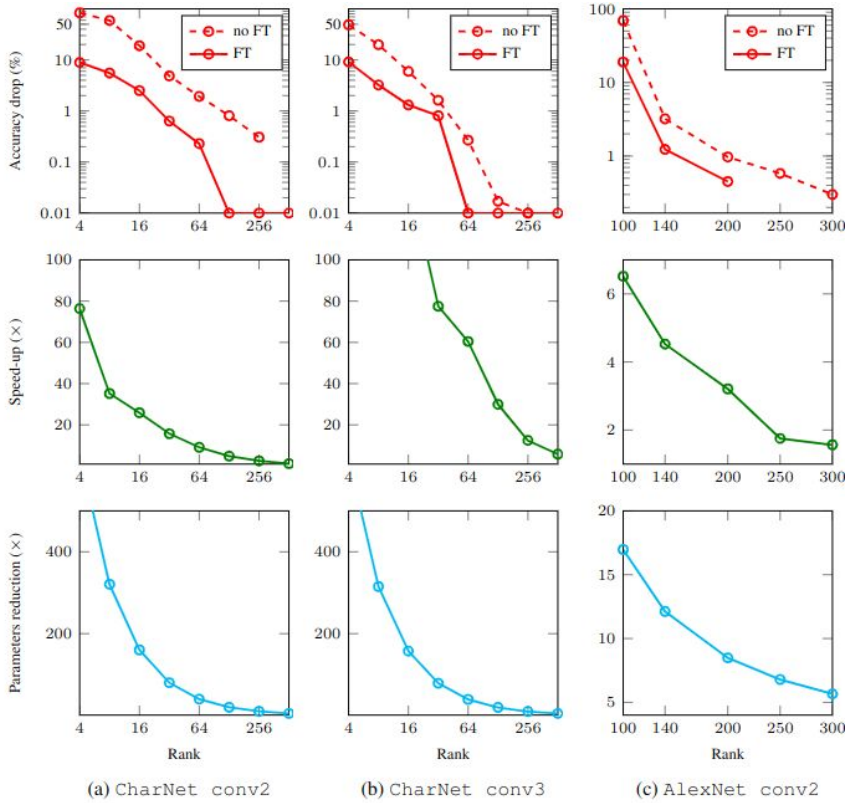
$$\boldsymbol{X} = \boldsymbol{a} \circledcirc \boldsymbol{b} = \boldsymbol{a}\boldsymbol{b}^T$$

$$\mathcal{X} = \boldsymbol{a}^{(1)} \circledcirc \boldsymbol{a}^{(2)} \circledcirc \cdots \circledcirc \boldsymbol{a}^{(N)} \text{ with } x_{i_1 i_2 \cdots i_N} = a^{(1)}_{i_1} a^{(2)}_{i_2} \cdots a^{(N)}_{i_N}$$



Convolution Speed up with CPD

- Применим CPD к весам в свертках
- Измерим
  - Ускорение
  - Просадку в качестве
  - Экономию в параметрах
- Везде выигрываем



Источник: Lebedev, Vadim, et al. "Speeding-up convolutional neural networks using fine-tuned cp-decomposition." arXiv preprint arXiv:1412.6553 (2014).

Как умножать тензоры

- Умножить тензор на матрицу = умножить совпадающую размерность проходясь по всем остальным размерностям
- Можно мысленно переставлять нужную размерность в конец
- Можно думать как о цикле for
- Или смириться с формальным определением

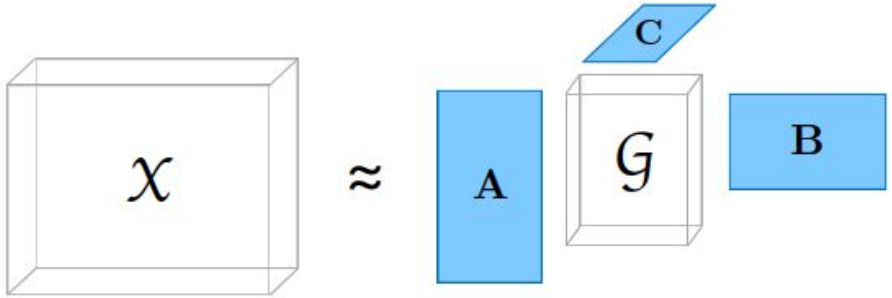
$$\mathcal{X} \in R^{I \times J \times K} \quad \mathcal{M} \in R^{J \times M}$$

$$(\mathcal{X} \times \mathcal{M}) \in R^{I \times N \times K}$$

$$(\mathcal{X} \times \mathcal{M})_{i,n,k} = \sum_{j=1}^{J} \mathcal{X}_{i,j,k} \mathcal{M}_{j,n}$$

Tucker Decomposition

- Один ключевой тензор
- Несколько отображений в размерности
- Наиболее близко по смыслу к Сингулярному

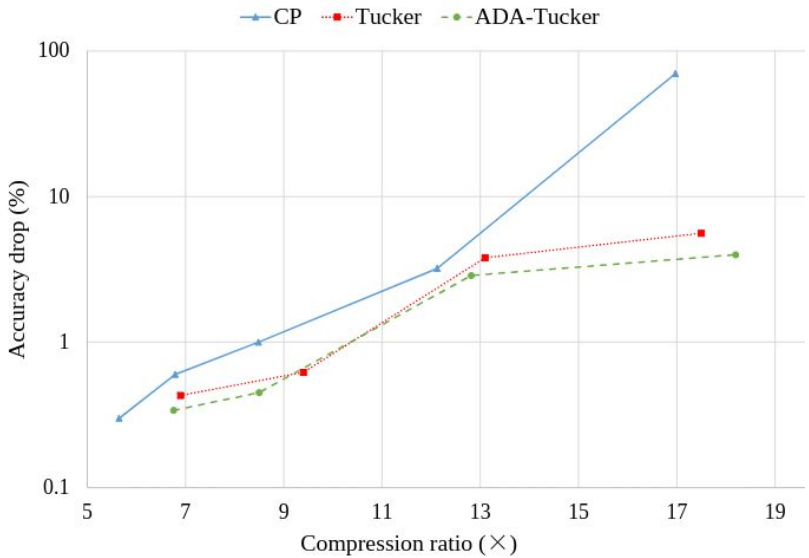


$$\mathcal{X} \in R^{I \times J \times K}$$

$$\mathcal{G} \in R^{P \times Q \times R} \quad A \in R^{I \times P} \quad B \in R^{J \times Q} \quad C \in R^{K \times R}$$

Tucker Decomposition for Convolutions

- Лучше чем CPD
- Дает высокое сжатие
- Можно сделать еще лучше маленькими эвристиками
- https://arxiv.org/pdf/1906.07671

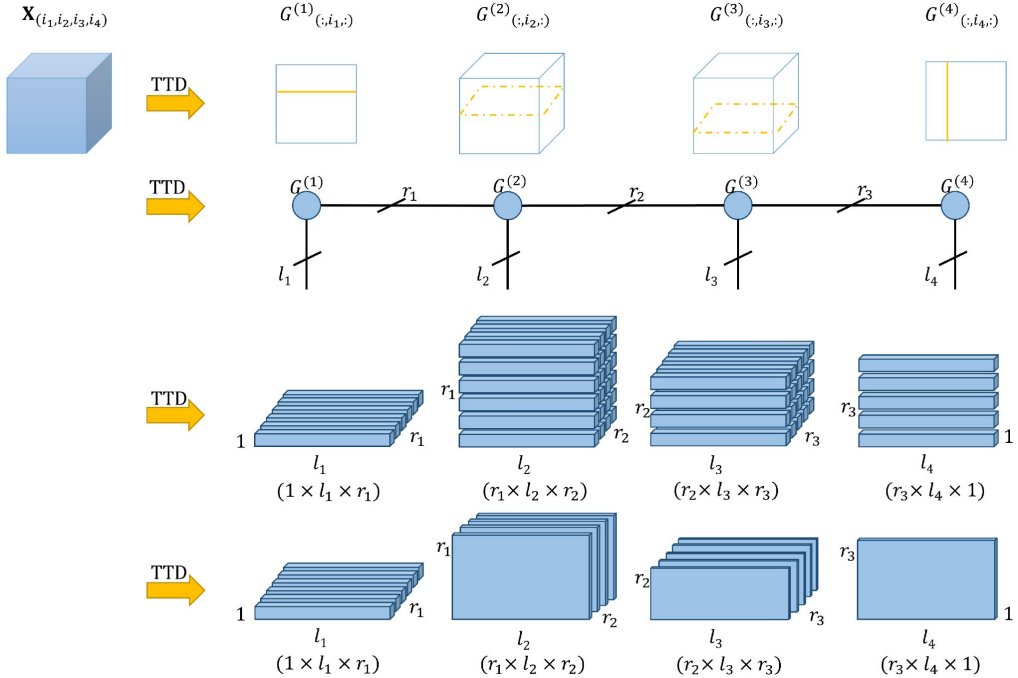


Network	#(Param.)	Orig. Acc.(%)	ADA-Tucker Acc.(%)	Δ(Acc.)	CR
ResNet-20	0.27M	91.25%	90.97%	-0.28%	**12**
WRN-28-10	36.5M	95.83%	95.06%	-0.77%	**58**

Источник: Zhong, Zhisheng, et al. "ADA-Tucker: Compressing deep neural networks via adaptive dimension adjustment tucker decomposition." Neural Networks 110 (2019): 104-115.

Tensor Train

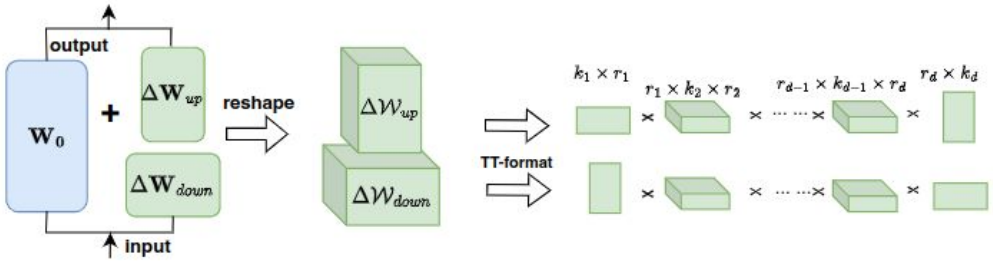
- Разложение в D трехмерных тензоров
- Обобщение тензорного ранга



Источник: Oseledets, Ivan V. "Tensor-train decomposition." SIAM Journal on Scientific Computing 33.5 (2011): 2295-2317.

Low-Rank Economic Tensor-Train Adaptation

- Решейп матрицы в тензор (обратная операция к векторизации)
- Разложение



Например:

- Матрица 768 x 64 = 49k
- Решейп в тензор (8 x 8 x 12 x 8 x 8)
- Разложение в Тензор Трейн с рангом 5 = 5 тензоров
  - 5 x 8 x 5 (2)
  - 5 x 8 x 1 (2)
  - 5 x 12 x 5 (1)
- Суммарное количество элементов = 780

Источник: Yang, Yifan, et al. "LoRETTA: Low-Rank Economic Tensor-Train Adaptation for Ultra-Low-Parameter Fine-Tuning of Large Language Models." arXiv preprint arXiv:2402.11417 (2024).

Low-Rank Economic Tensor-Train Adaptation

- В среднем лушче LoRA
- Энкодерная модель

Model & Method	# Train. Param.	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg.
DeBERTa-Base (FT)	139.19M	88.67	94.61	91.98	59.32	93.04	91.42	68.23	91.10	84.79
DeBERTa-Base (Adapters$_{r=8}$)	0.94M	87.69	94.72	88.88	54.19	92.95	85.52	59.20	89.68	81.60
DeBERTa-Base (LoRA$_{r=8}$)	0.30M	87.30	94.95	92.84	60.56	93.35	85.19	80.14	90.13	85.56
DeBERTa-Base (P-Tuning)	0.23M	56.25	91.39	79.93	43.31	86.30	78.43	55.95	78.38	71.24
DeBERTa-Base (LoRA$_{r=4}$)	0.15M	**87.69**	94.49	91.10	62.57	92.60	87.30	69.67	91.12	84.54
DeBERTa-Base (Prompt)	0.01M	77.63	92.43	81.90	32.99	80.30	78.15	62.81	56.71	70.36
DeBERTa-Base (Prefix)	0.15M	60.32	88.87	81.22	45.82	83.28	82.22	59.57	84.99	73.28
DeBERTa-Base (BitFit)	0.10M	84.63	95.41	91.42	**64.06**	93.30	84.15	66.79	90.23	83.75
DeBERTa-Base (LoRETTA$_{adp}$)	0.10M	85.93	95.30	**93.53**	60.84	92.99	84.08	75.50	**91.32**	**84.96**
DeBERTa-Base (LoRETTA$_{rep}$)	0.05M	86.80	**95.53**	88.73	59.69	**93.25**	**89.2**	**75.81**	90.66	84.95
RoBERTa-Base (BitFit) *	0.1M	85.30	**94.80**	**92.33**	62.70	91.30	68.10	73.60	88.50	82.08
RoBERTa-Base (LoRA$_{r=8}$)*	0.63M	**86.82**	94.01	91.48	62.08	**92.39**	85.71	74.51	**90.48**	84.69
RoBERTa-Base (LoRETTA$_{adp}$)	0.10M	85.61	94.38	91.08	**62.70**	92.12	**87.22**	**78.70**	90.26	**85.26**

>>> Спасибо за внимание!