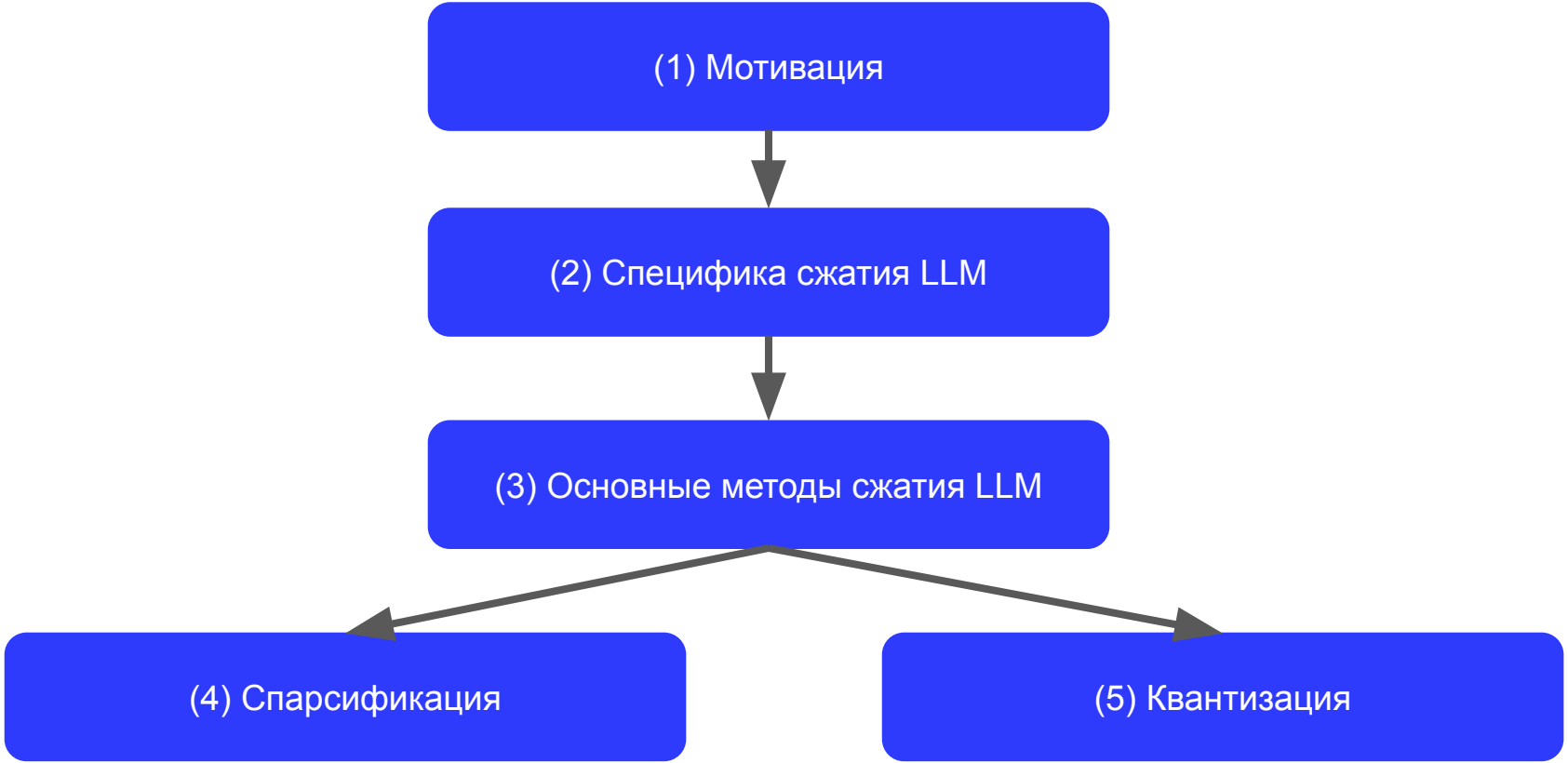
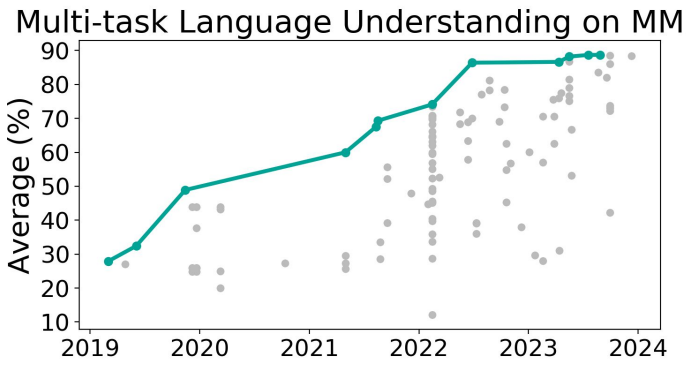
Методы сжатия больших языковых моделей

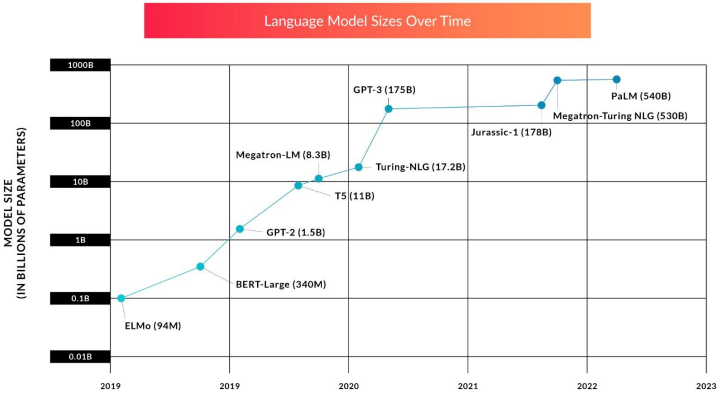
Кузнеделев Денис


(1) Мотивация
(2) Специфика сжатия LLM
(3) Основные методы сжатия LLM
(4) Спарсификация
(5) Квантизация

Зачем сжимать LLM?



Большие языковые модели становятся все умнее и умнее

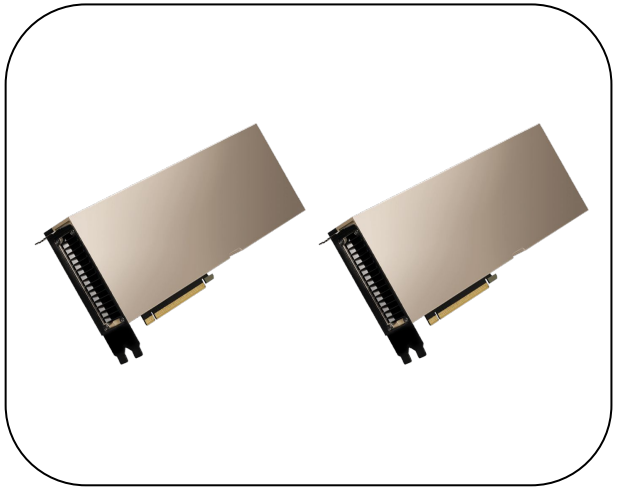


Но и больше, увы…

Зачем сжимать LLM?

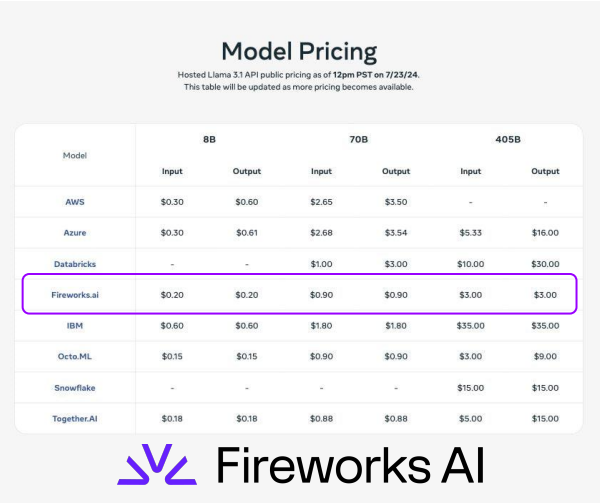
Llama-3-70b в fp16/bf16

=



2x Nvidia A100

Кому это нужно?



Model Pricing

Hosted Llama 3.1 API public pricing as of **12pm PST on 7/23/24.**
This table will be updated as more pricing becomes available.

Model	8B		70B		405B	
	Input	Output	Input	Output	Input	Output
AWS	$0.30	$0.60	$2.65	$3.50	-	-
Azure	$0.30	$0.61	$2.68	$3.54	$5.33	$16.00
Databricks	-	-	$1.00	$3.00	$10.00	$30.00
Fireworks.ai	$0.20	$0.20	$0.90	$0.90	$3.00	$3.00
IBM	$0.60	$0.60	$1.80	$1.80	$35.00	$35.00
Octo.ML	$0.15	$0.15	$0.90	$0.90	$3.00	$9.00
Snowflake	-	-	-	-	$15.00	$15.00
Together.AI	$0.18	$0.18	$0.88	$0.88	$5.00	$15.00

Fireworks AI

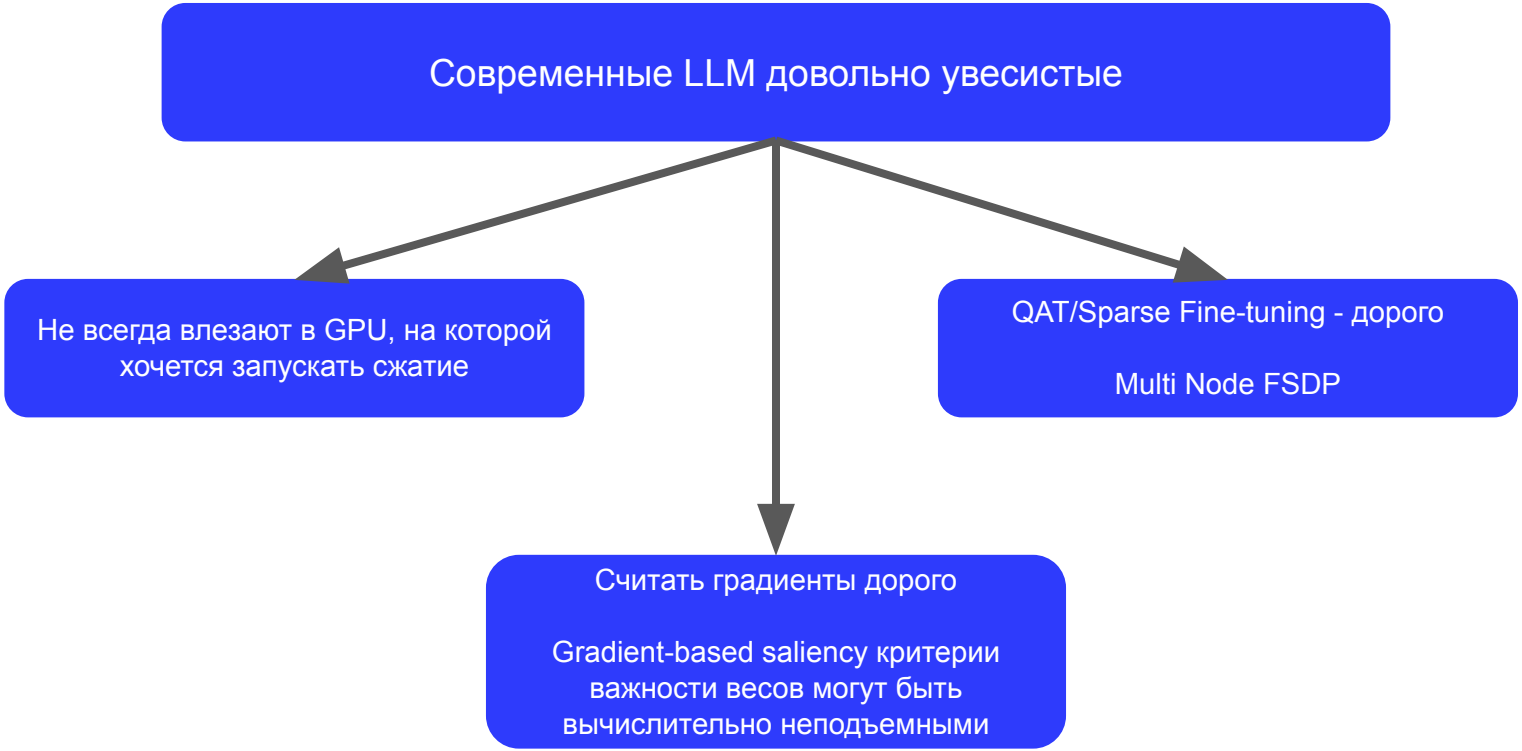
Провайдерам для снижения стоимости инференса.

GGUF

turboderp/
exllamav2

Optimum
Documentation

Простым смертным для инференса на пользовательском железе.


Современные LLM довольно увесистые
Не всегда влезают в GPU, на которой хочется запускать сжатие
QAT/Sparse Fine-tuning - дорого
Multi Node FSDP
Считать градиенты дорого
Gradient-based saliency критерии важности весов могут быть вычислительно неподъемными

Современные LLM довольно увесистые

Методы сжатия делятся на следующие категории

Data-Free

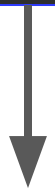
LLM.int8()
HQQ
bnb-4bit

Data-aware с локальными критериями

GPTQ
AWQ
SparseGPT
QuIP#
AQLM
Efficient QAT

Инференс LLM Memory-bound

Основное время занимают не сами вычисления, а время на подрузку весов модели из HDM/GDDR в кэши GPU



Можно добиться ускорения
за счет сжатия одних лишь весов, не трогая активации

Optimal Brain Surgeon

Нейронная сеть, вообще говоря, сложная нелинейная функция.

Для упрощения описания в окрестности данной точки прибегают к разложению лосс-функции до **2-го порядка** по формуле Тейлора:

$$\mathcal{L}(\mathbf{w}) - \mathcal{L}(\mathbf{w}^*) \simeq \nabla_{\mathcal{L}}^{\top}(\mathbf{w}^*)(\mathbf{w} - \mathbf{w}^*) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^*)^{\top}\mathbf{H}_{\mathcal{L}}(\mathbf{w}^*)(\mathbf{w} - \mathbf{w}^*)$$

Функция потерь	**Член 1-го порядка**	**Член 2-го порядка**
	Равен 0 в точке оптимума	Наш бро

Optimal Brain Surgeon

Проиллюстрируем метод на примере прунинга одного веса $\mathbf{w}_i$

$$\min_{\mathbf{w}}(\mathcal{L}(\mathbf{w}) - \mathcal{L}(\mathbf{w}^*)) \quad \text{при условии} \quad \mathbf{w}_i + \mathbf{e}_i^\top \mathbf{w} = 0$$

Имеет следующее решение

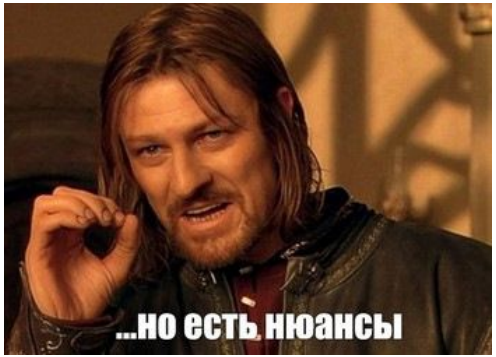
$$\rho_i = \frac{w_i^2}{2[\mathbf{H}_{\mathcal{L}}^{-1}(\mathbf{w}^*)]_{ii}}$$

Приращение ошибки

$$\delta \mathbf{w}^* = -\frac{w_i}{[\mathbf{H}_{\mathcal{L}}^{-1}(\mathbf{w}^*)]_{ii}} \mathbf{H}_{\mathcal{L}}^{-1}(\mathbf{w}^*)\mathbf{e}_i,$$

Оптимальный сдвиг остальных весов

Optimal Brain Surgeon



Гессиан квадратичен по количеству параметров в сети - **очень-очень** дорого

Optimal Brain Surgeon

Решение?

Использовать ошибку на выходе данного слоя

$$\text{argmin}_{\widehat{\mathbf{W}}_\ell} \quad ||\mathbf{W}_\ell \mathbf{X}_\ell - \widehat{\mathbf{W}}_\ell \mathbf{X}_\ell||_2^2$$

1) Память, требуемая для хранения Гессиана $\Theta(d_{col}^2)$
2) Гессиан один и тот же для всех выходных каналов слоя
3) В процессе сжатия не нужно загружать всю модель в память

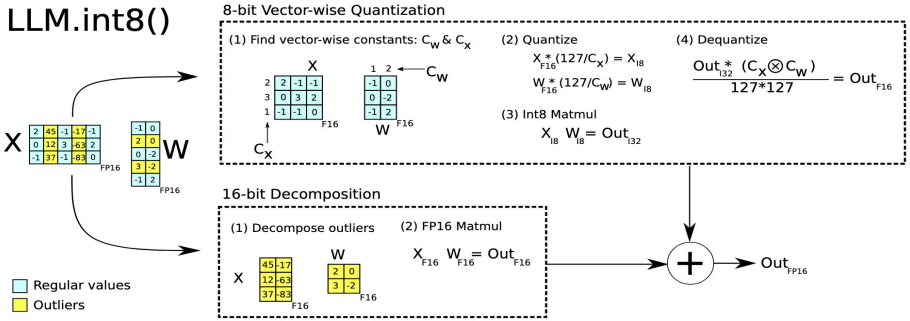
GPTQ SpQR Wanda AQLM

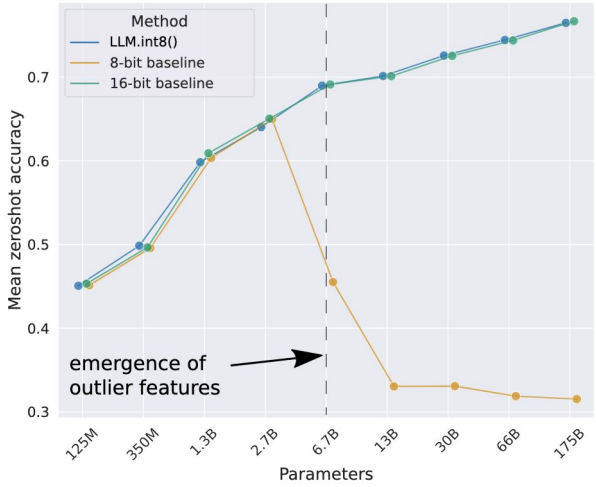
SparseGPT AWQ QuIP#

Выбросы / Outliers

В многих современных нейронных сетях есть небольшая доля весов, крайне чувствительных к изменениям.

В работе LLM.int8() было предложено не трогать их при квантизации и держать в исходной точности.





Из LLM.int8()

Dettmers, Tim, et al. "Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale." *Advances in neural information processing systems* 35 (2022): 30318-30332.

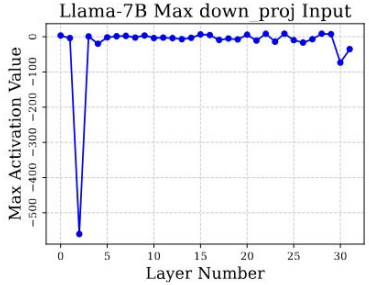
Выбросы / Outliers

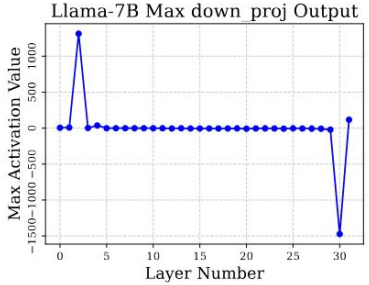
Супер-веса - отдельные параметры, с огромным влиянием на выход сети

Model	No.	Type	Weight	Coordinates
Llama 7B	2	mlp	down_proj	[3968, 7003]
Llama 13B	2 2	mlp mlp	down_proj down_proj	[2231, 2278] [2231, 6939]
Llama 30B	3 3 10	mlp mlp mlp	down_proj down_proj down_proj	[5633, 12817] [5633, 17439] [5633, 14386]
Llama2 7B	1	mlp	down_proj	[2533, 7890]
Llama2 13B	3	mlp	down_proj	[4743, 7678]
Mistral-7B v0.1	1	mlp	down_proj	[2070, 7310]

Model	No.	Type	Weight	Coordinates
OLMo-1B 0724-hf	1 1	mlp mlp	down_proj down_proj	[1764, 1710] [1764, 8041]
OLMo-7B 0724-hf	1 2 7 24	mlp mlp mlp mlp	down_proj down_proj down_proj down_proj	[269, 7467] [269, 8275] [269, 453] [269, 2300]
Phi-3 mini-4k-instruct	2 2 2 4 4 4	mlp mlp mlp mlp mlp mlp	down_proj down_proj down_proj down_proj down_proj down_proj	[525, 808] [1693, 808] [1113, 808] [525, 2723] [1113, 2723] [1693, 2723]

Table 2: Super Weight Directory. The above layer numbers, layer types, and weight types can be directly applied to Huggingface models. For example, for Llama-7B on Huggingface, access the super weight using `layers[2].mlp.down_proj.weight[3968, 7003]`.





Yu, Mengxia, et al. "The super weight in large language models." *arXiv preprint arXiv:2411.07191* (2024).

Хороший метод сжатия LLM

Масштабируемый

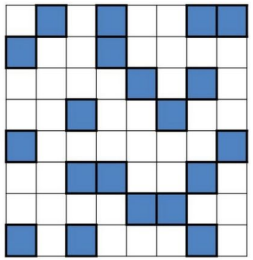
Учитывающий выбросы

С эффективными кернелами…

Спарсификация

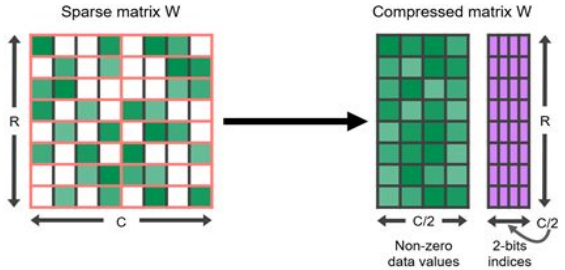
Неструктурированная

Нет ускорения на GPU :(



Полуструктурированная

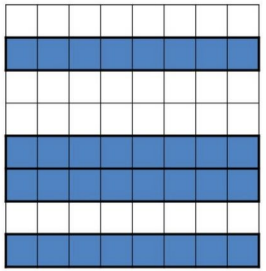
Ускорение на NVIDIA Ampere+



SparseGPT
Wanda
ADMM-Pruning

Структурированная

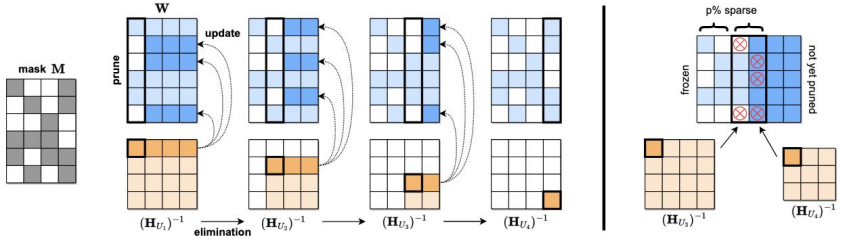
Почти гарантированное ускорение



ZipLM
CoFi
LLMPruner

SparseGPT

1) Использует послойное приближение
2) Убирает веса с наименьшей ошибкой
3) Для эффективного обновления Гессиана сжатого веса используется Cholesky разложение.



Визуализация алгоритма

Algorithm 1 The `SparseGPT` algorithm. We prune the layer matrix $\mathbf{W}$ to $p\%$ unstructured sparsity given inverse Hessian $\mathbf{H}^{-1} = (\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I})^{-1}$, lazy batch-update block-size B and adaptive mask selection blocksize B_s; each B_s consecutive columns will be $p\%$ sparse.

$\mathbf{M} \leftarrow \mathbf{1}_{d_{\text{row}} \times d_{\text{col}}}$ *// binary pruning mask*
$\mathbf{E} \leftarrow \mathbf{0}_{d_{\text{row}} \times B}$ *// block quantization errors*
$\mathbf{H}^{-1} \leftarrow \text{Cholesky}(\mathbf{H}^{-1})^\top$ *// Hessian inverse information*
for $i = 0, B, 2B, \ldots$ **do**
  for $j = i, \ldots, i + B - 1$ **do**
    if $j \bmod B_s = 0$ **then**
      $\mathbf{M}_{:,j:(j+B_s)} \leftarrow$ mask of $(1-p)\%$ weights $w_c \in \mathbf{W}_{:,j:(j+B_s)}$ with largest $w_c^2 / [\mathbf{H}^{-1}]_{cc}^2$
    end if
    $\mathbf{E}_{:,j-i} \leftarrow \mathbf{W}_{:,j} \,/\, [\mathbf{H}^{-1}]_{jj}$ *// pruning error*
    $\mathbf{E}_{:,j-i} \leftarrow (\mathbf{1} - \mathbf{M}_{:,j}) \cdot \mathbf{E}_{:,j-i}$ *// freeze weights*
    $\mathbf{W}_{:,j:(i+B)} \leftarrow \mathbf{W}_{:,j:(i+B)} - \mathbf{E}_{:,j-i} \cdot \mathbf{H}^{-1}_{j,j:(i+B)}$ *// update*
  end for
  $\mathbf{W}_{:,(i+B):} \leftarrow \mathbf{W}_{:,(i+B):} - \mathbf{E} \cdot \mathbf{H}^{-1}_{i:(i+B),(i+B):}$ *// update*
end for
$\mathbf{W} \leftarrow \mathbf{W} \cdot \mathbf{M}$ *// set pruned weights to 0*

Алгоритм

Frantar, Elias, and Dan Alistarh. "Sparsegpt: Massive language models can be accurately pruned in one-shot." *International Conference on Machine Learning*. PMLR, 2023.

SparseGPT

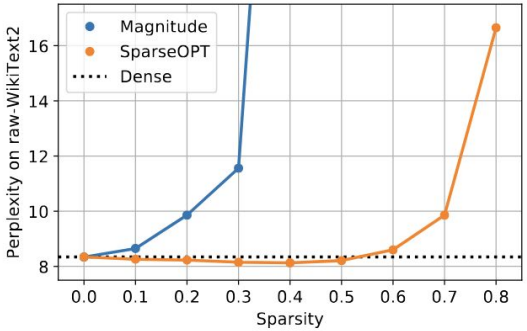


Figure 1. Sparsity-vs-perplexity comparison of `SparseGPT` against magnitude pruning on OPT-175B, when pruning to different *uniform* per-layer sparsities.

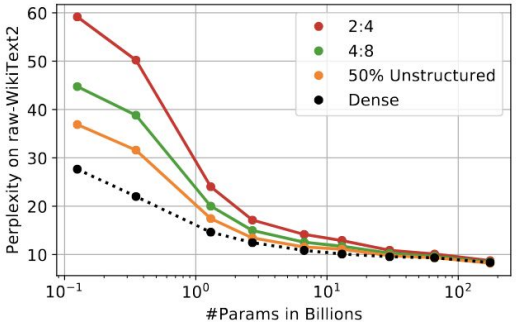


Figure 2. Perplexity vs. model and sparsity type when compressing the entire OPT model family (135M, 350M, ..., 66B, 175B) to different sparsity patterns using `SparseGPT`.

Работает лучше Data-free

Большие модели легче сжимать

Frantar, Elias, and Dan Alistarh. "Sparsegpt: Massive language models can be accurately pruned in one-shot." *International Conference on Machine Learning*. PMLR, 2023.

Wanda

Ограничимся диагональю Гессиана

$$\mathbf{S}_{ij} = |\mathbf{W}_{ij}| \cdot \|\mathbf{X}_j\|_2$$

Критерий важности

Вполне себе работает…

			LLaMA				LLaMA-2		
Method	Weight Update	Sparsity	7B	13B	30B	65B	7B	13B	70B
Dense	-	0%	5.68	5.09	4.77	3.56	5.12	4.57	3.12
Magnitude	✗	50%	17.29	20.21	7.54	5.90	14.89	6.37	4.98
SparseGPT	✓	50%	**7.22**	6.21	5.31	**4.57**	6.51	5.63	**3.98**
Wanda	✗	50%	7.26	**6.15**	**5.24**	**4.57**	**6.42**	**5.56**	**3.98**
Magnitude	✗	4:8	16.84	13.84	7.62	6.36	16.48	6.76	5.54
SparseGPT	✓	4:8	8.61	**7.40**	6.17	5.38	8.12	6.60	4.59
Wanda	✗	4:8	**8.57**	**7.40**	**5.97**	**5.30**	**7.97**	**6.55**	**4.47**
Magnitude	✗	2:4	42.13	18.37	9.10	7.11	54.59	8.33	6.33
SparseGPT	✓	2:4	**11.00**	**9.11**	7.16	6.28	**10.17**	8.32	5.40
Wanda	✗	2:4	11.53	9.58	**6.90**	**6.25**	11.02	**8.27**	**5.16**

Table 3: WikiText perplexity of pruned LLaMA and LLaMA-2 models. Wanda performs competitively against prior best method SparseGPT, without introducing any weight update.

Sun, Mingjie, et al. "A simple and effective pruning approach for large language models." *arXiv preprint arXiv:2306.11695* (2023).

ZipLM

1) Использует послойное приближение
2) Убирает входную размерность / группу размерностей с наименьшей ошибкой за один ход.
3) В контексте сжатия трансформеров ZipLM за один шаг убирает **hidden_dim в FFN** или целую **attention голову**.
4) Gradual pruning с дообучением.

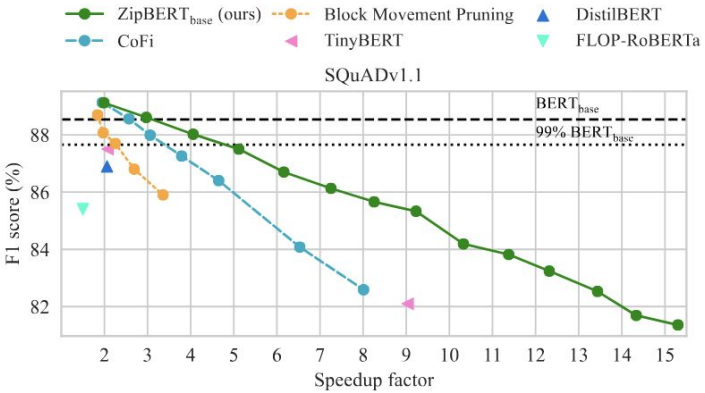
Algorithm 1 The ZipLM pruning algorithm. Given inverse Hessian $\mathbf{H}^{-1} = (2\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I})^{-1}$, we remove exactly k structures from the corresponding weight matrix $\mathbf{W}$.

$\mathbf{R} \leftarrow$ set of all possible structures
for k times **do**
  $\mathbf{S} \leftarrow \text{argmin}_{\mathbf{S}} \sum_{i=0}^{d_{\text{row}}} \mathbf{W}_{i,\mathbf{M_S}} \cdot ((\mathbf{H}^{-1})_{\mathbf{M_S},\mathbf{M_S}})^{-1} \cdot \mathbf{W}_{i,\mathbf{M_S}}^\top$
  $\boldsymbol{\delta}_S \leftarrow -\mathbf{W}_{:,\mathbf{M_S}} \cdot ((\mathbf{H}^{-1})_{\mathbf{M_S},\mathbf{M_S}})^{-1} \cdot (\mathbf{H}^{-1})_{\mathbf{M_S},:}$
  $\mathbf{W} \leftarrow \mathbf{W} + \boldsymbol{\delta}_S$
  $\mathbf{H}^{-1} \leftarrow \mathbf{H}^{-1} - \mathbf{H}^{-1}_{:,\mathbf{M_S}} \cdot ((\mathbf{H}^{-1})_{\mathbf{M_S},\mathbf{M_S}})^{-1} \cdot \mathbf{H}^{-1}_{\mathbf{M_S},:}$
  $\mathbf{R} \leftarrow \mathbf{R} - \{\mathbf{S}\}$
end for
$\mathbf{W} \leftarrow \mathbf{W} \odot \mathbf{M}_{\mathbf{R}}$

Алгоритм

Kurtić, Eldar, Elias Frantar, and Dan Alistarh. "Ziplm: Inference-aware structured pruning of language models." *Advances in Neural Information Processing Systems* 36 (2023): 65597-65617.

ZipLM



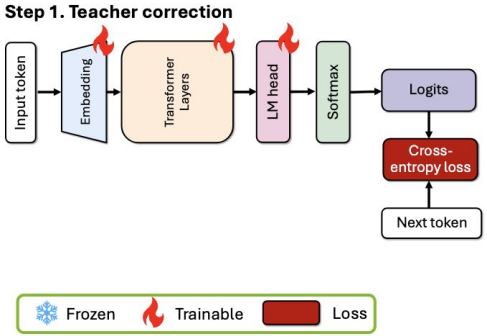
BERT хорошо сжимается

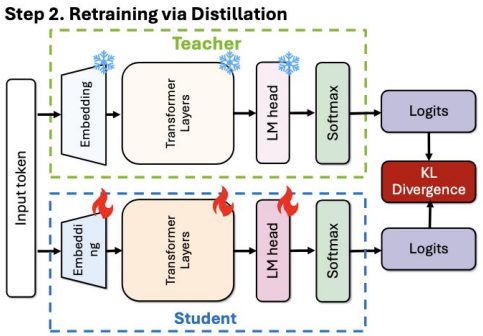
Model	Pruning for throughput			Pruning for latency		
	Speedup	Decoder size	Wiki Text-103 PPL ↓	Speedup	Decoder size	Wiki Text-103 PPL ↓
GPT2*	1.0x	85.0M	28.5	1.0x	85.0M	28.5
DistilGPT2	**1.6x**	**42.5M**	**43.0**	**1.9x**	**42.5M**	**43.0**
ZipGPT2 (ours)	**1.5x**	**47.3M**	**35.4**	**1.6x**	**48.7M**	**37.8**
	2.1x	**26.5M**	**41.5**	**2.0x**	**39.2M**	**41.2**
	2.7x	14.0M	50.4	2.2x	26.6M	49.0
	3.3x	5.7M	72.1	2.5x	20.7M	55.0

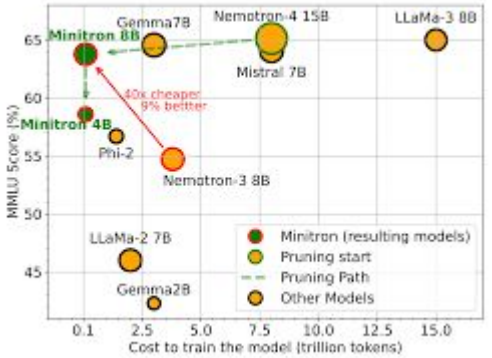
Гопота не очень :(

Kurtić, Eldar, Elias Frantar, and Dan Alistarh. "Ziplm: Inference-aware structured pruning of language models." *Advances in Neural Information Processing Systems* 36 (2023): 65597-65617.

Minitron







Если дистиллировать сжатую модель на большом количестве данных (~500B токенов) можно выжать хорошее качество.

Sreenivas, Sharath Turuvekere, et al. "Llm pruning and distillation in practice: The minitron approach." *arXiv preprint arXiv:2408.11796* (2024).

Спарсификация

На практике редко удается добиться существенного сжатия/ускорения без значительной просадки в качестве или длительного до-обучения

Квантизация - способ уменьшения количества памяти, требуемой на хранение массива данных, за счет представления его элементов из некоторого ограниченного множества.

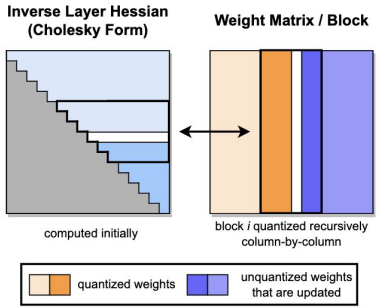
fp16/bf16 - 16 бит на параметр
INT8/fp8 - 8 бит на параметр
INT4/fp4* - 4 бит на параметр

GPTQ

1) Использует послойное приближение
2) Для эффективного обновления Гессиана сжатого веса используется Cholesky разложение.

Algorithm 1 Quantize $\mathbf{W}$ given inverse Hessian $\mathbf{H}^{-1} = (2\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I})^{-1}$ and blocksize B.

$\mathbf{Q} \leftarrow \mathbf{0}_{d_{\text{row}} \times d_{\text{col}}}$ *// quantized output*
$\mathbf{E} \leftarrow \mathbf{0}_{d_{\text{row}} \times B}$ *// block quantization errors*
$\mathbf{H}^{-1} \leftarrow \text{Cholesky}(\mathbf{H}^{-1})^\top$ *// Hessian inverse information*
for $i = 0, B, 2B, \ldots$ **do**
  for $j = i, \ldots, i + B - 1$ **do**
    $\mathbf{Q}_{:,j} \leftarrow \text{quant}(\mathbf{W}_{:,j})$ *// quantize column*
    $\mathbf{E}_{:,j-i} \leftarrow (\mathbf{W}_{:,j} - \mathbf{Q}_{:,j}) / [\mathbf{H}^{-1}]_{jj}$ *// quantization error*
    $\mathbf{W}_{:,j:(i+B)} \leftarrow \mathbf{W}_{:,j:(i+B)} - \mathbf{E}_{:,j-i} \cdot \mathbf{H}^{-1}_{j,j:(i+B)}$ *// update weights in block*
  end for
  $\mathbf{W}_{:,(i+B):} \leftarrow \mathbf{W}_{:,(i+B):} - \mathbf{E} \cdot \mathbf{H}^{-1}_{i:(i+B),(i+B):}$ *// update all remaining weights*
end for



[1] Frantar, Elias, et al. "Gptq: Accurate post-training quantization for generative pre-trained transformers." *arXiv preprint arXiv:2210.17323* (2022).

GPTQ

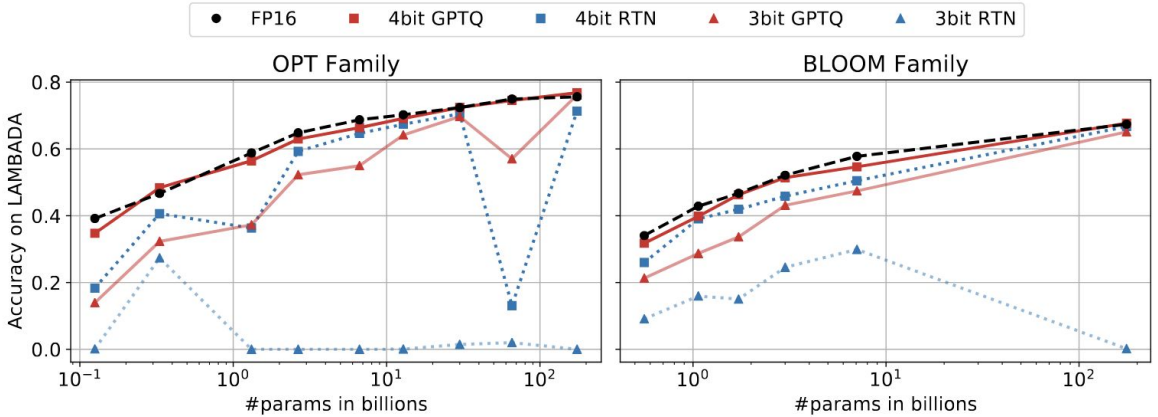


Figure 3: The accuracy of OPT and BLOOM models post-GPTQ, measured on LAMBADA.

GPTQ сжимает почти* без просадки LLM

Frantar, Elias, et al. "Gptq: Accurate post-training quantization for generative pre-trained transformers." *arXiv preprint arXiv:2210.17323* (2022).

AWQ

масштаб нуль

$$w_q = S(q + Z)$$

Представление веса при квантизации

$$\mathbf{s}^* = \arg\min_{\mathbf{s}} \mathcal{L}(\mathbf{s})$$

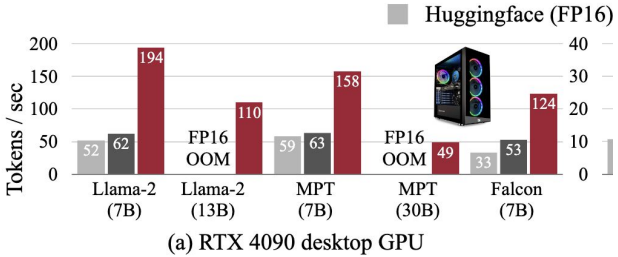
$$\mathcal{L}(\mathbf{s}) = \|Q(\mathbf{W} \cdot \mathrm{diag}(\mathbf{s}))(\mathrm{diag}(\mathbf{s})^{-1} \cdot \mathbf{X}) - \mathbf{W}\mathbf{X}\|$$

Задача оптимизации

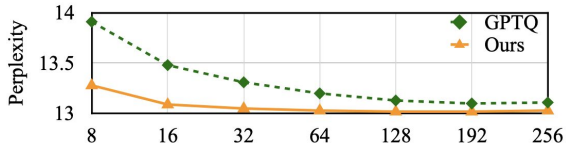
1) Оптимизирует Scale для послойной ошибки
2) Оптимизация осуществляется через поиск по сетке
3) На инференсе надо делить на выученный масштаб - недорого

Lin, Ji, et al. "Awq: Activation-aware weight quantization for on-device llm compression and acceleration." *Proceedings of Machine Learning and Systems* 6 (2024): 87-100.

AWQ



(a) RTX 4090 desktop GPU



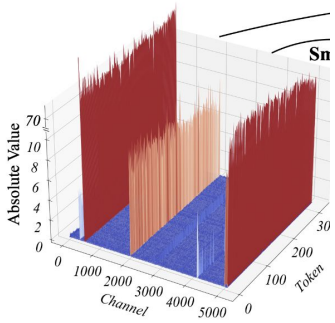
(a) Our method needs a smaller calibration set

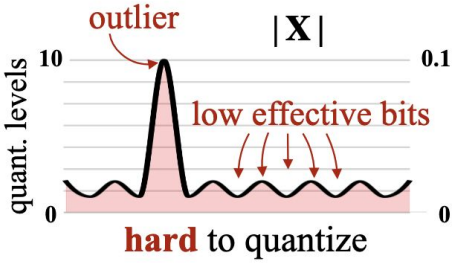
PPL↓		Llama-2		
		7B	13B	70B
FP16	-	5.47	4.88	3.32
INT3 g128	RTN	6.66	5.52	3.98
	GPTQ	6.43	5.48	3.88
	GPTQ-R	6.42	5.41	3.86
	AWQ	**6.24**	**5.32**	**3.74**
INT4 g128	RTN	5.73	4.98	3.46
	GPTQ	5.69	4.98	3.42
	GPTQ-R	5.63	4.99	3.43
	AWQ	**5.60**	**4.97**	**3.41**

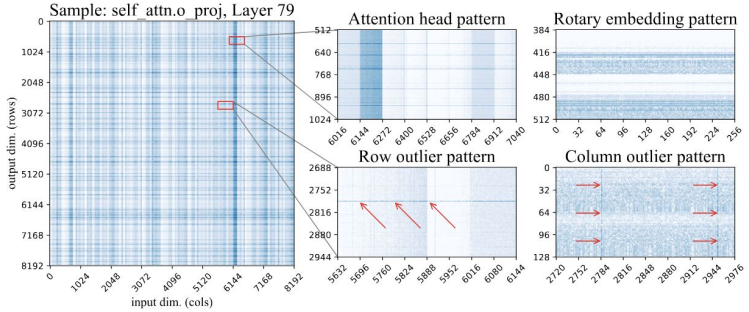
1) Дает неплохое ускорение инференса
2) Примерно по качеству равен GPTQ (если не жульничать :))
3) Требует меньшего количества сэмплов для сходимости

Lin, Ji, et al. "Awq: Activation-aware weight quantization for on-device llm compression and acceleration." *Proceedings of Machine Learning and Systems* 6 (2024): 87-100.

Учет выбросов







Изоляция и хранение в виде разреженной матрицы

LLM.int8()
SpQR
SqueezeLLM

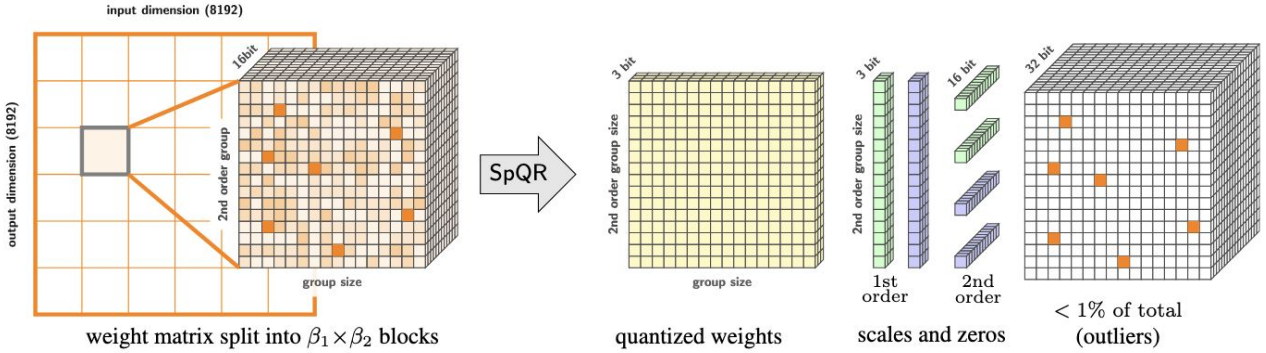
Переход в другой базис

QuIP
QuIP#
QuaRot

SpQR

$$s_{ij} = \frac{(w_{ij} - \text{quant}(w_{ij}))^2}{2(XX^\top)^{-1}}$$

Метрика для определения важных весов



1) “Важные” веса не квантизуются
2) Для большего сжатия скейлы тоже квантизуются

Dettmers, Tim, et al. "Spqr: A sparse-quantized representation for near-lossless llm weight compression." *arXiv preprint arXiv:2306.03078* (2023).

QuIP

Algorithm 1 QuIP - Incoherence Pre-Processing

Require: $b \in \mathbb{N}$, $H \in \mathbb{R}^{n\times n}$ SPD, original $W \in \mathbb{R}^{m\times n}$, $\rho \in \mathbb{R}_+$, $\alpha \in [0,1]$

1: **seeded sample** random two-factor orthogonal matrices $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$

2: $H = H + \alpha * \text{mean}(\text{diag}(H))I$ ▷ from OPTQ

3: $\tilde{D} \leftarrow \sqrt[4]{\text{diag}(H)/\text{diag}(W^TW)}$ ▷ $\sqrt[4]{\cdot}$ applies element-wise

4: $W \leftarrow W\tilde{D}$; $H \leftarrow \tilde{D}^{-1}H\tilde{D}^{-1}$ ▷ diagonal rescaling

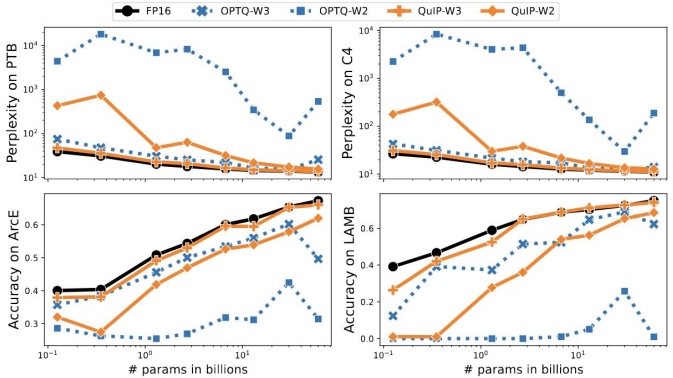
5: $W \leftarrow UWV^T$; $H \leftarrow VHV^T$ ▷ incoherence

6: $s \leftarrow \rho\|W\|_F/\sqrt{mn}$; $W \leftarrow \frac{1}{2}(\frac{1}{s}W + 1)$ ▷ reduced quantization range due to incoherency

7: $W \leftarrow \text{clamp}(W * (2^b - 1), 0, 2^b - 1)$ ▷ rescale W to lie within $[0, 2^b - 1]$

8: **return** $W, H, s, \tilde{D}$

Алгоритм



Сеть не разлетается при 2-битной квантизации

1) Поворот весов на случайную ортогональную матрицу “регуляризует” распределение весов
2) Матрицы поворота надо применять к активациям на инференсе
3) Матрицы поворота довольно компактные и “дешевые” (matvec быстрее, чем O(n^2))

Chee, Jerry, et al. "Quip: 2-bit quantization of large language models with guarantees." *Advances in Neural Information Processing Systems* 36 (2023): 4396-4429.

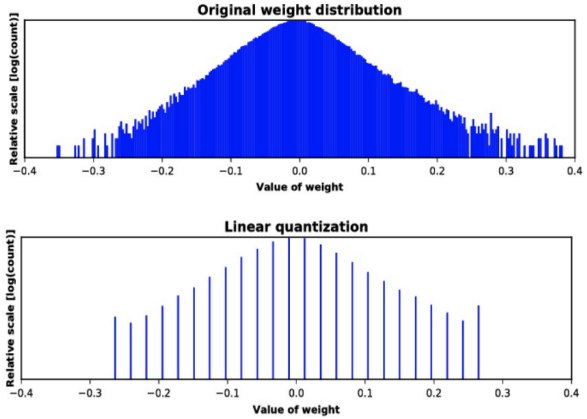
Скалярная и векторная квантизация

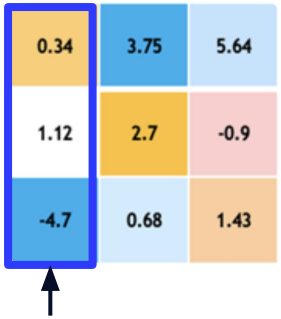
Скалярная

каждому значению сопоставляется индекс на 1-мерной сетке

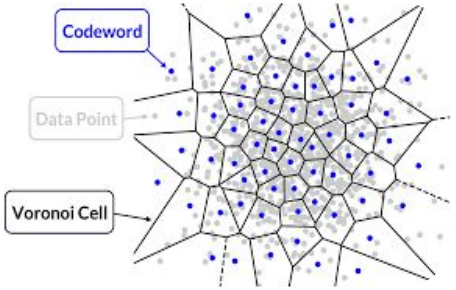
Векторная

каждой **группе** значений сопоставляется **вектор** из **кодовой** книги





квантизуемые совместно веса

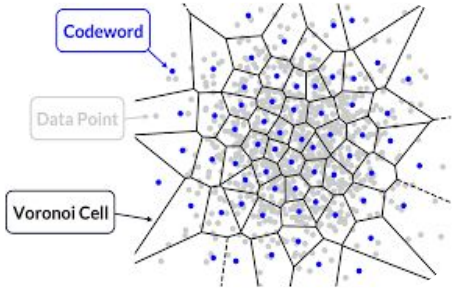


Скалярная и векторная квантизация

Векторная

каждой **группе** значений сопоставляется **вектор** из **кодовой** книги

Кодовые слова - векторы (узлы)
Кодовая книга - совокупность кодовых слов (решетка)
Код - индекс вектора в решетке



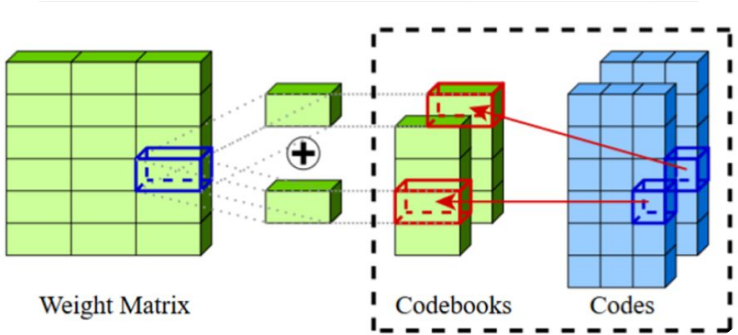
AQLM

1) Представим группу подряд идущих весов как взвешенную сумму векторов из одной или более кодовых книг
2) Используем послойное приближение
3) Аддитивная квантизация возникла изначально в контексте задачи оптимизации поиска

$$\sum_{m=1}^{M} C_m b_m$$

C - кодовые слова

b - one-hot коды



Представление веса

Egiazarian, Vage, et al. "Extreme compression of large language models via additive quantization." *arXiv preprint arXiv:2401.06118* (2024).

AQLM

Как найти оптимальные C и b?

Целевая функция

$$||\mathbf{WX} - \sum_{m=1}^{M} C_m b_m \mathbf{X}||_2^2$$

1) Непрерывные параметры (C) оптимизируются через градиентный спуск (Adam)
2) Дискретные параметры (b) оптимизируются с помощью beam search
3) Оптимизирует C и b поочередно

На практике нужно 10^2-10^3 итераций для сходимости

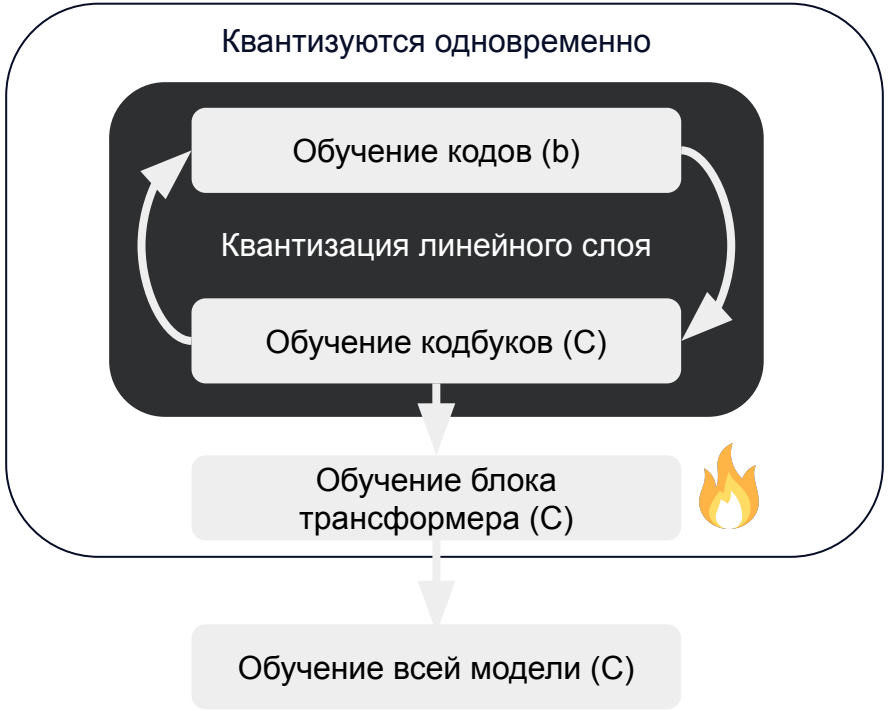
AQLM

Как найти оптимальные C и b?

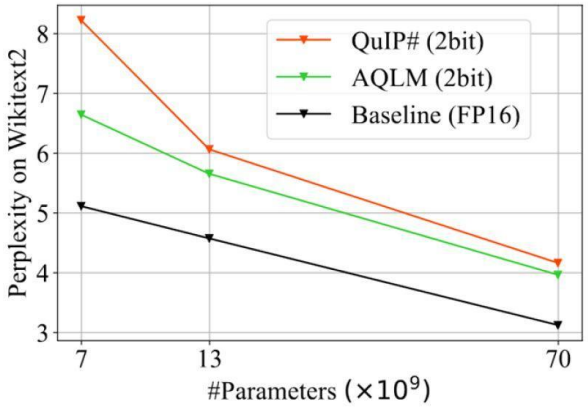
Конечная цель не послойная ошибка - а итоговое качество
Можно ли оптимизировать более “глобально”?

- Для блочной оптимизации используется **L2** между признаками исходной и сжатой модели
- Для end-2-end (следуя QuIP#) - **KL-дивергенция**

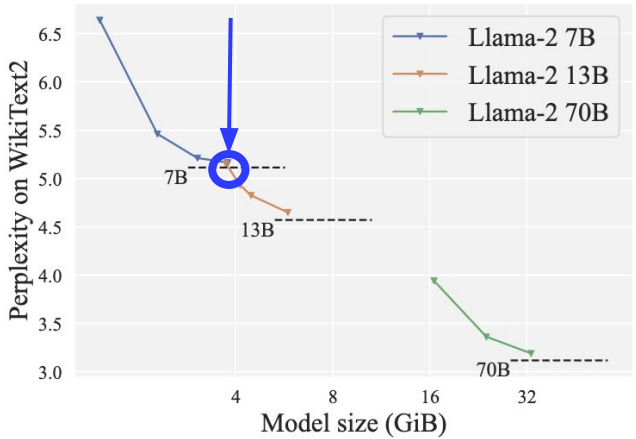
AQLM


Квантизуются одновременно
Обучение кодов (b)
Квантизация линейного слоя
Обучение кодбуков (C)
Обучение блока трансформера (C)
Обучение всей модели (C)

AQLM



Результаты при 2-битной квантизации



Парето-оптимальность при 2.5 битах

Парето-оптимальность
- оптимальный размер модели и степень сжатия при заданном ограничении на память

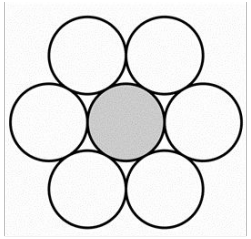
QuIP#

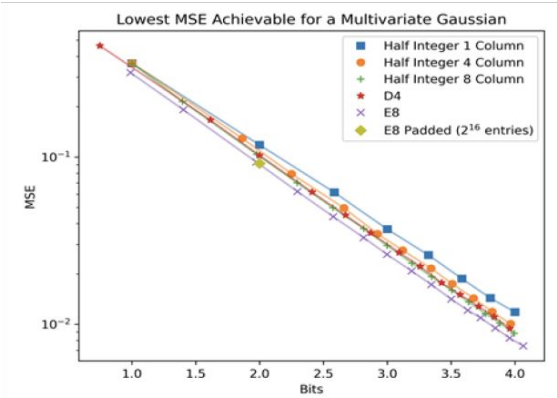
1) Конкурентный подход векторной квантизации.
2) Используется послойное приближение.
3) Вместо обучаемой решетки используется решетка для **самой плотной упаковки** сфер в 8-мерном пространстве.
4) Перед квантизацией веса поворачиваются с помошью Адамаровых матриц.
5) Приводит почти любое распределение весов к **нормальному**.

$$H_0 = +(1)$$

$$H_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

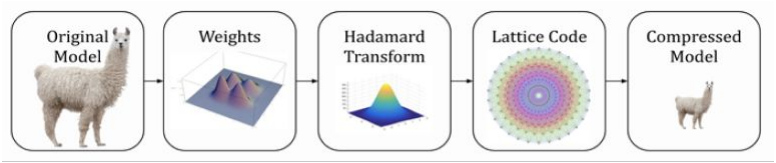
$$H_2 = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$





Tseng, Albert, et al. "Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks." *arXiv preprint arXiv:2402.04396* (2024).

QuIP#

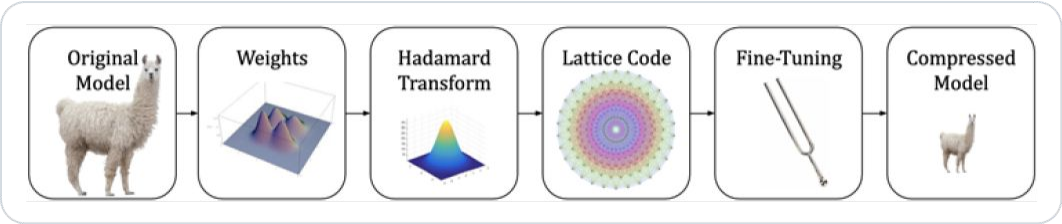


Quantization results on Llama 2 70B. QuIP# achieves near-native performance at 2 bits, outperforming all other presented baselines.

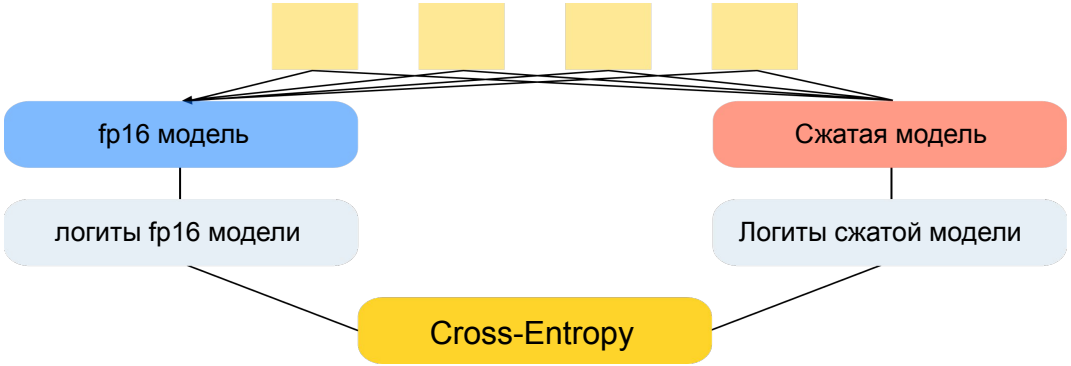
Method	Precision	Wiki ↓	C4 ↓	ArcE ↑	PiQA ↑
Native	16 bit	3.120	5.533	0.597	0.809
OPTQ	3 bit	4.577	6.838	0.544	**0.786**
OPTQ	2 bit	109.820	62.692	0.253	0.505
QuIP	2 bit	5.574	8.268	0.544	0.751
QuIP#	**2 bit**	**4.159**	**6.529**	**0.595**	0.786

Вполне себе SOTA при 2-битной квантизации

QuIP#



Дистилляция с исходной моделью накидывает качество

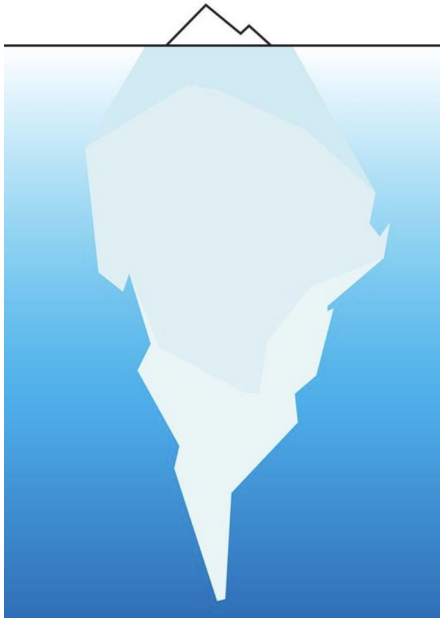


PV-Tuning

После квантизации большинство параметров дискретные и не могут быть обучаться через GD

Большинство методов оптимизируют end-2-end только непрерывные параметры

Оптимизация дискретных параметров имеет большой потенциал



Непрерывные параметры
10-100M

Дискретные параметры
1-100B

Malinovskii, Vladimir, et al. "Pv-tuning: Beyond straight-through estimation for extreme llm compression." *Advances in Neural Information Processing Systems* 37 (2024): 5074-5121.

PV-Tuning

Algorithm 6 PV algorithm

1: **Initialization:** starting point $x^0 \in \mathbb{R}^d_{\leq c}$
2: **for** $k = 0, 1, \ldots$ **do**
3: $\quad y^k = M_P(x^k) := \arg\min_{y \in \mathbb{R}^d} \left\{ \phi(y) : P(y) \supseteq P(x^k) \right\}$ (P step: continuous)
4: $\quad x^{k+1} = M_V(y^k) := \arg\min_{x \in \mathbb{R}^d} \left\{ \phi(x) : V(x) \subseteq V(y^k) \right\}$ (V step: discrete)
5: **end for**

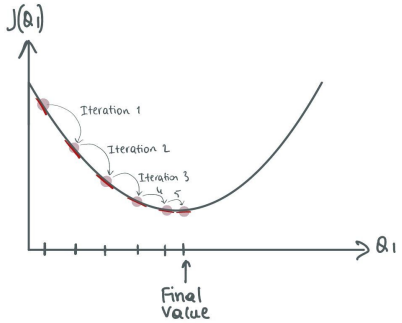
Дискретные и непрерывные параметры можно обучать через альтернированную оптимизацию

P - partition (дискретные)
V - values (непрерывные)

PV-Tuning

Как оптимизировать **непрерывные** и **дискретные** параметры?

Непрерывные



Любой алгоритм градиентного спуска

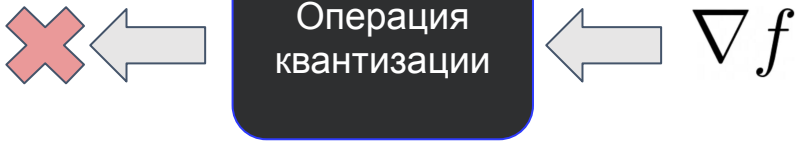
Дискретные

[1] https://towardsdatascience.com/understanding-gradient-descent-for-machine-learning-246e324c229

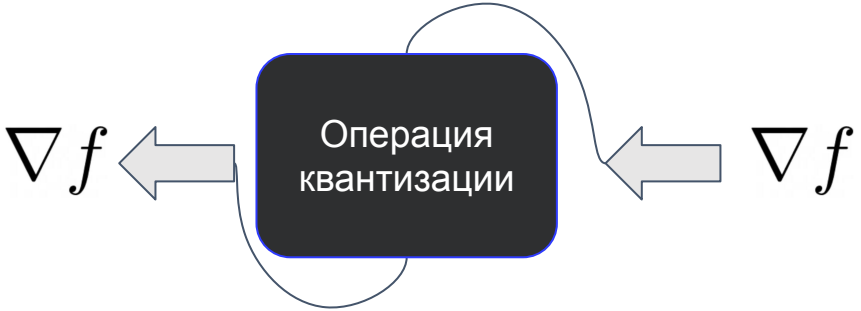
PV-Tuning

Операция квантизации **не дифференцируема**

Градиентный спуск



Straight Through Estimator (STE):
copy/paste градиент



Bengio, Yoshua, Nicholas Léonard, and Aaron Courville. "Estimating or propagating gradients through stochastic neurons for conditional computation." *arXiv preprint arXiv:1308.3432* (2013).

PV-Tuning

Наивный STE **нестабилен**

Малые шаги

Оптимизация застревает

Большие шаги

Оптимизация улетает в космос

PV-Tuning

Идея:
Обновлять только небольшую долю (подпространство) параметров с самой **большой нормой** градиента

Это дает **устойчивую** и **быструю** сходимость

Algorithm 7 PV-Tuning: Optimization

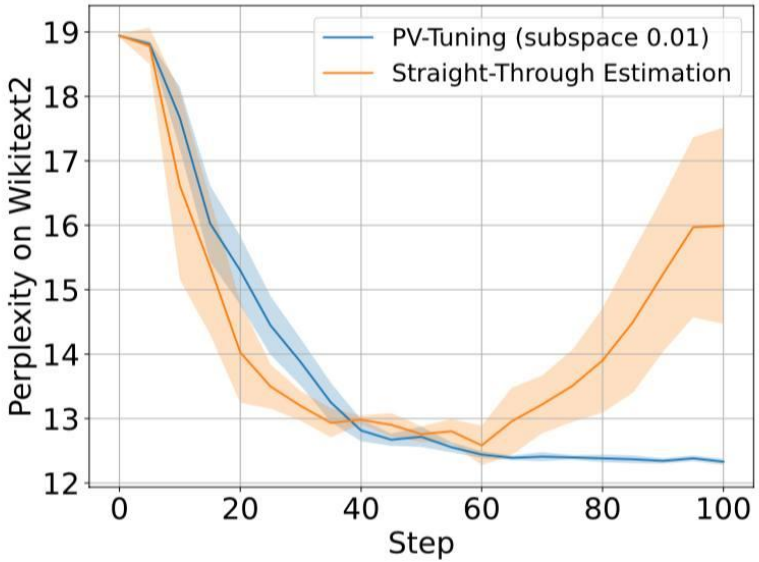
initial parameters $x^0 \in \mathbb{R}^d_c$,
objective function $\phi : \mathbb{R}^d \to \mathbb{R}$,
subspace size $\tau \in [d]$
for $k = 0, \ldots, K-1$ **do**
  ▷ **P step:** update $V(x)$ by backprop
  $y^k = \underset{y \in R^d_{\leq c}}{arg\,min}\{\phi(y) : P(y) \supseteq P(x^k)\}$
  ▷ **V step:** choose a subspace $\mathcal{S}^k$ & update $P(x)$
  $\mathcal{S}^k = \underset{1 \leq i \leq d}{arg\,top}\,\tau \quad |\nabla_i \phi(y^k)|$ ▷ find τ largest
  $\widehat{\phi}_{y,\mathcal{S}^k}(x) := \left\| x - \left(y - \frac{1}{L_{\mathcal{S}^k}} Z^k \left(\nabla \phi(y) \right) \right) \right\|^2$, Z^k - subspace projection operator
  $x^{k+1} \in \arg\min_x \left\{ \widehat{\phi}_{y^k,\mathcal{S}^k}(x) : V(x) \subseteq V(y^k) \right\}$
end for

Алгоритм

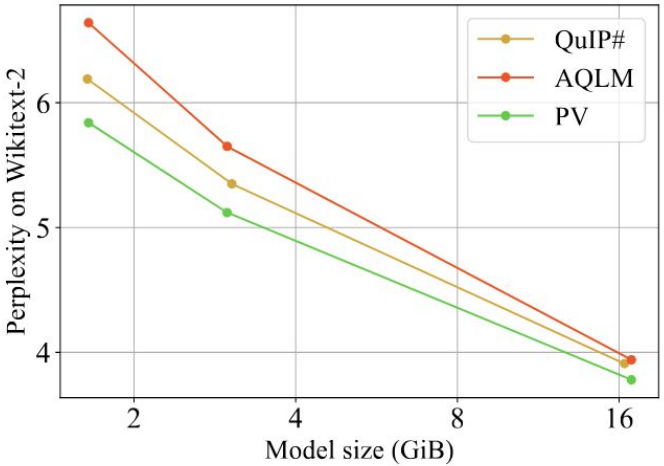
На каждом **V-шаге** (дискретная оптимизация) малая доля кодов обновляется с помощью **STE**

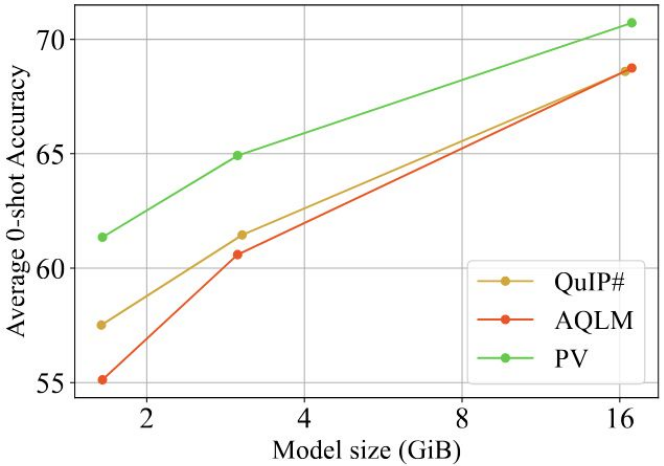
PV-Tuning

Эффект от ограничения на обновляемые параметры



PV-Tuning





PV-Tuning позволяет сильно улучшить качество по сравнению с базовым алгоритмом квантизации.

PV-Tuning

Fine-tuning Method	GPTQ, 2.14 bit/w			VQ, 1.58 bit/w			AQLM, 2.01 bit/w		
	Wiki2↓	C4↓	Acc.↑	Wiki2↓	C4↓	Acc.↑	Wiki2↓	C4↓	Acc.↑
Calibration only (no global fine-tuning)	3290	4125	29.0	20.26	20.09	43.42	7.38	9.34	53.2
Continuous params only [1, 3]	16.77	17.53	46.27	8.17	10.99	52.14	6.69	8.77	56.57
Naive Linearized PV (no subspace)	16.73	17.48	47.68	8.19	10.94	52.08	6.68	8.75	56.51
Stochastic Rounding(tuned) [2]	11.97	13.07	49.79	8.02	10.64	52.31	6.56	8.39	56.68
Straight Through Estimation [4]	8.79	11.04	50.61	7.76	10.26	52.58	6.41	8.63	57.04
Subspace Linearized PV (ours, τ=0.01)	8.49	**10.78**	**52.17**	7.38	9.47	53.36	6.13	8.35	57.81
Subspace Linearized PV+STE (τ=0.01)	**8.43**	10.82	51.90	**7.32**	**9.35**	**55.22**	**5.90**	**7.43**	**58.19**

PV-Tuning совместим с разными алгоритмами квантизации

HIGGS

1) Применяем к матрице весов преобразование Адамара
2) Отображаем на оптимальную n-мерную сетку для Гауссово-распределенных весов
3) Совместимо с data-aware и data-free методами

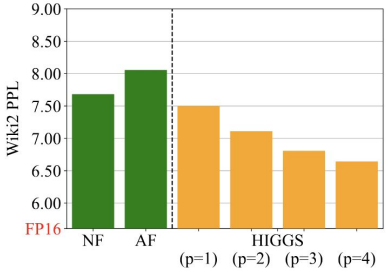


Figure 2: Comparison of Normal Float (NF), Abnormal Float (AF) and HIGGS on Llama 3.1 8B quantization to 3.19-3.25 bitwidth range. HIGGS is instantiated at different lattice dimensionalities p.

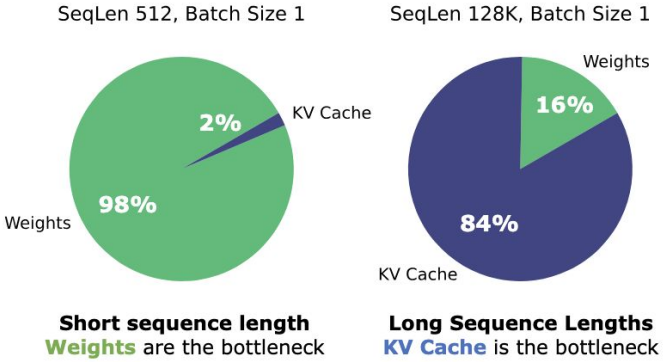
Table 2: WikiText-2 PPL comparison of various 1-shot quantization methods for Llama-2-7b.

FP16		GPTQ	AQLM	QuIP#	QTIP	GPTQ+HIGGS ($p = 2$)
5.117	wbits≈2	-	8.180	8.220	6.820	8.637
	wbits≈3	5.776	-	5.600	5.380	5.559
	wbits≈4	5.254	-	5.220	5.170	5.213

Malinovskii, Vladimir, et al. "Pushing the limits of large language model quantization via the linearity theorem." *arXiv preprint arXiv:2411.17525* (2024).

Сжатие KV-кэшей

Для инференса Causal LLM обыкновенно сохраняют Key, Value для прошлых токенов, чтобы не пересчитывать каждый раз.



Для длинных последовательностей - KV-кэш может занимать памяти больше, чем сама модель.
Кроме того, длинные кэши замедляют инференс - основное время уходит на выгрузку кэшей в кэш GPU.

Hooper, Coleman, et al. "Kvquant: Towards 10 million context length llm inference with kv cache quantization." *Advances in Neural Information Processing Systems* 37 (2024): 1270-1303.

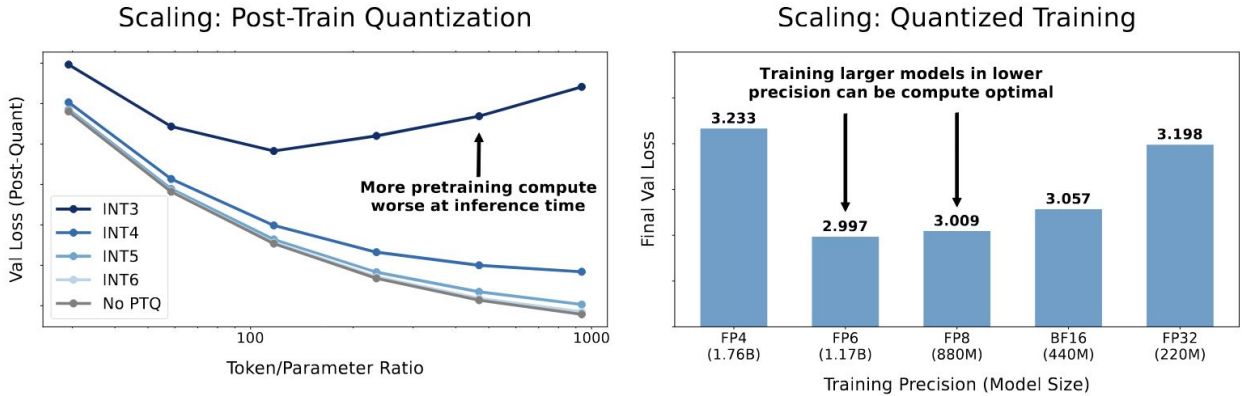
Сжатие KV-кэшей

Существуют разнообразные решения для уменьшения памяти на хранение KV-кэшей

Архитектурные	Прунинг кэшей	Квантизация
Grouped Query Attention	H2O	KIVI
Multihead Latent Attention	SnapKV	KVQuant
Native Sparse Attention	PyramidKV	AQUA-KV

Scaling Laws for Precision

Как зависит качество модели от бюджета обучения и битности параметров?

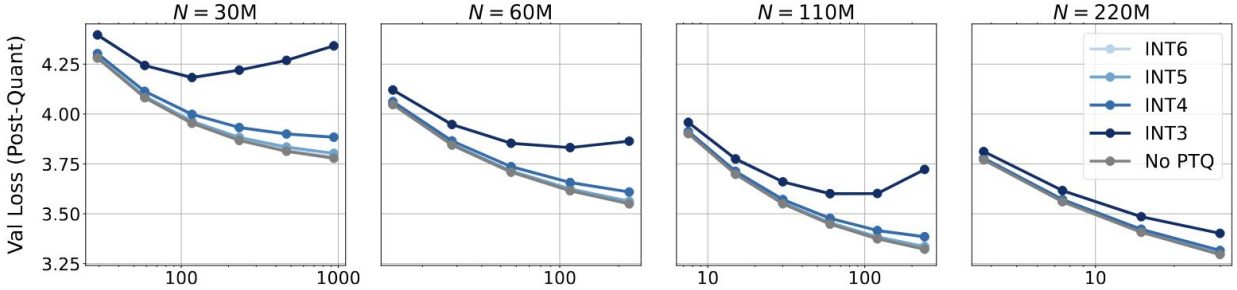


Рассматривают два сценария квантизации:

1) Post-Train (квантизация после обучения)
2) Quantized Training (квантизация в процессе обучения)

Scaling Laws for Precision

Post-Train



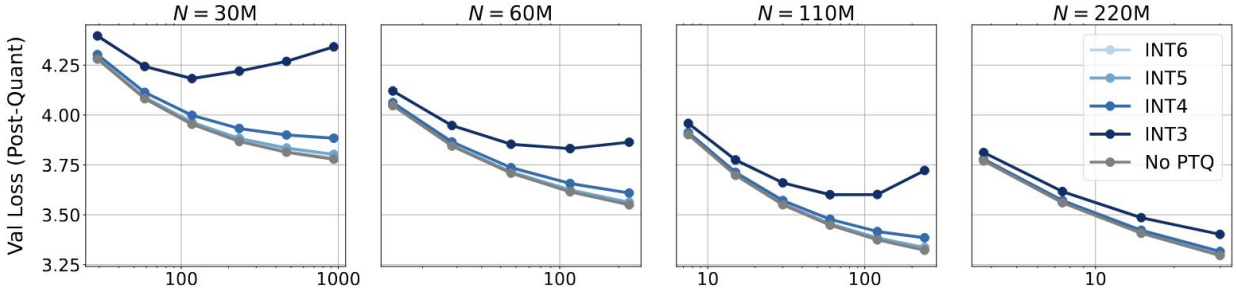
Модели, которых дольше обучали, страдают сильнее от квантизации

$$\delta_{\text{PTQ}}(N, D, P_{\text{post}}) = C_T \left(\frac{D^{\gamma_D}}{N^{\gamma_N}} \right) e^{-P_{\text{post}}/\gamma_{\text{post}}}$$

Прирост лосса при квантизации описывается неплохо зависимостью выше

Scaling Laws for Precision

Post-Train



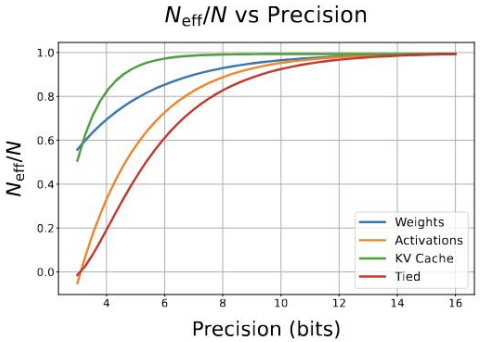
Модели, которых дольше обучали, страдают сильнее от квантизации

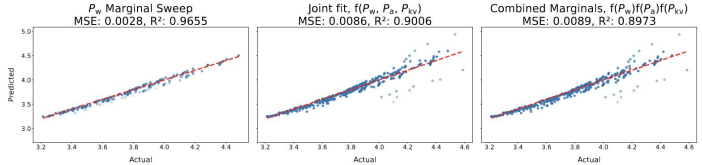
$$\delta_{\text{PTQ}}(N, D, P_{\text{post}}) = C_T \left(\frac{D^{\gamma_D}}{N^{\gamma_N}} \right) e^{-P_{\text{post}}/\gamma_{\text{post}}}$$

Прирост лосса при квантизации описывается неплохо зависимостью выше

Scaling Laws for Precision

Quantized Training





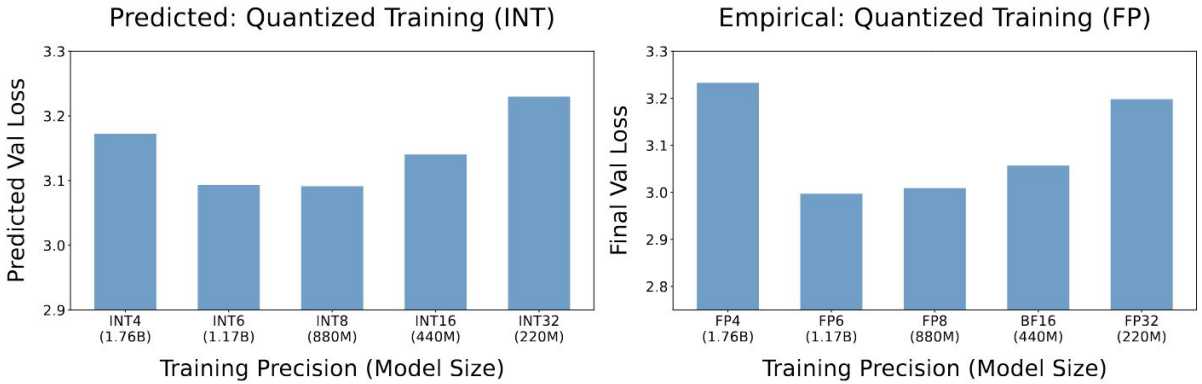
Фитирование коэффициентов на конфигурациях с разным размером модели/бюджетом обучения и точностью

Эффект от квантизации весов/активаций и KV-кэшей можно абсорбировать в “эффективное” количество параметров модели

$$N_{\text{eff}}(P) = N(1 - e^{-P_{\text{w}}/\gamma_{\text{w}}})(1 - e^{-P_{\text{a}}/\gamma_{\text{a}}})(1 - e^{-P_{\text{kv}}/\gamma_{\text{kv}}})$$

Scaling Laws for Precision

Quantized Training



Валидационный лосс при фиксированном размере модели

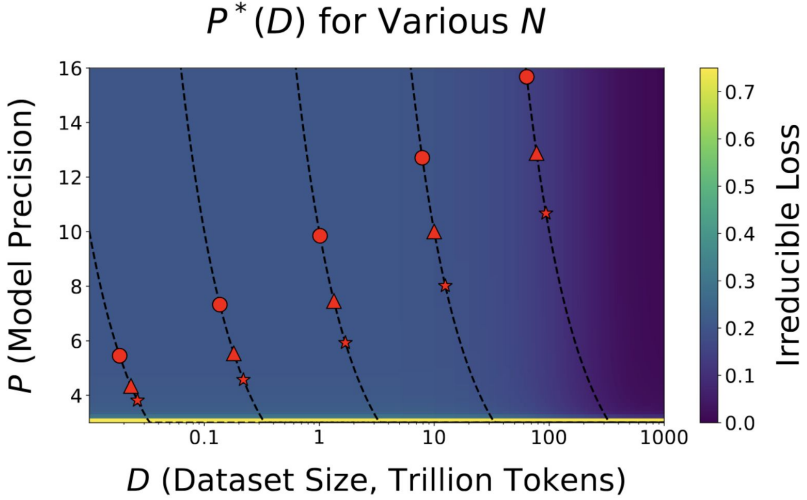
Scaling Laws for Precision

Quantized Training

8B

70B

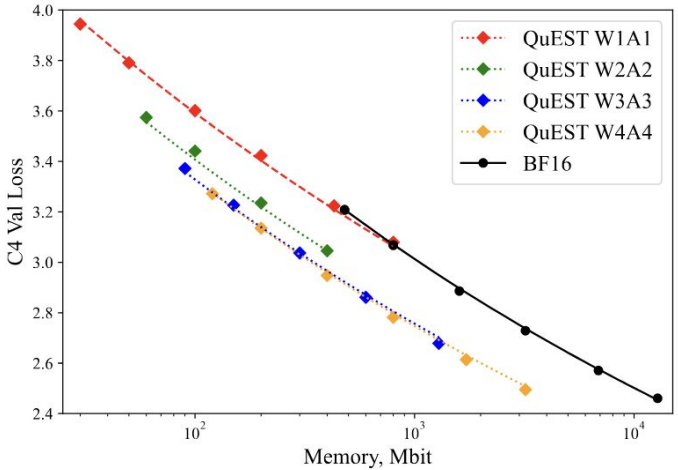
405B



Оптимальная точность при фиксированном размере модели

QuEST

Стабильного обучения можно добиться даже для 1-битных весов и активаций!



Лосс на C4 против размера моделей для разных битностей

QuEST

Но как?

$$\widehat{\mathbf{x}} = \alpha^* \cdot \mathrm{RMS}(\mathbf{x}) \cdot \left\lfloor \frac{\mathrm{clip}\left(\mathbf{x}/\mathrm{RMS}(\mathbf{x}), \alpha^*\right)}{\alpha^*} \right\rceil =$$

$$:= \mathrm{proj}_{\alpha^*}(\mathbf{x}), \text{ where}$$

$$\alpha^* := \arg\min_{\alpha \in \mathbb{R}} \mathbb{E}_{\xi \sim \mathcal{N}(0,1)} \left\| \xi - \alpha \cdot \left\lfloor \frac{\mathrm{clip}(\xi, \alpha)}{\alpha} \right\rceil \right\|_2^2$$

Пре-нормализация
весов

Algorithm 1 `QuEST Training Forward`

Input: Input activations $\mathbf{x}$, row-major weight $\mathbf{w}$
$\mathbf{x}_h = \mathrm{HT}(\mathbf{x})$
$\hat{\mathbf{x}}_h = \mathrm{proj}_{\alpha^*}\, \mathbf{x}_h$
$\mathbf{w}_h = \mathrm{HT}(\mathbf{w})$
$\hat{\mathbf{w}}_h = \mathrm{proj}_{\alpha^*}\, \mathbf{w}_h$
$\mathbf{y} = \hat{\mathbf{x}}_h \hat{\mathbf{w}}_h^T$
Return: $\mathbf{y}$, $\hat{\mathbf{x}}_h$, $\hat{\mathbf{w}}_h$, $M_{\alpha^*}(\mathbf{x}_h; \hat{\mathbf{x}}_h)$, $M_{\alpha^*}(\mathbf{w}_h; \hat{\mathbf{w}}_h)$

“Обработка“
Адамаровыми матрицами

$$\frac{\partial}{\partial \mathbf{x}} \approx \mathrm{IHT}\left(M_{\alpha^*}(\mathbf{x}_h; \hat{\mathbf{x}}_h) \odot \frac{\partial}{\partial \hat{\mathbf{x}}_h} \right).$$

Маскирование
“шумных” градиентов

QuEST

Но как?

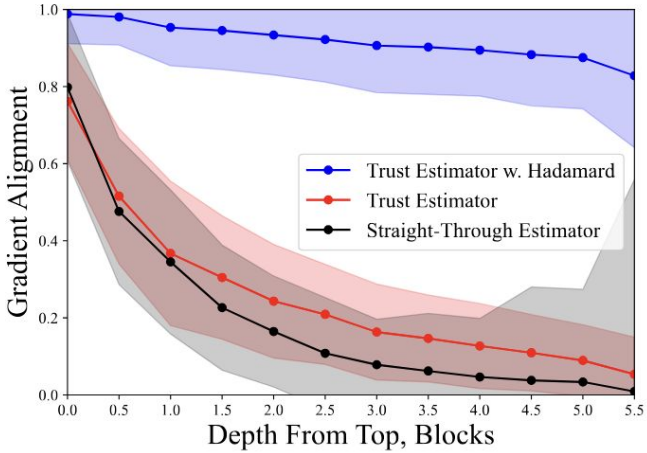


Figure 2. Gradient alignment comparison for a 30M Llama model after training on 2.7B tokens in 8-bit precision.

Маскирование и обработка матрицами важны!

QuEST

Table 1. Fitted scaling-law “effective parameter” multipliers.

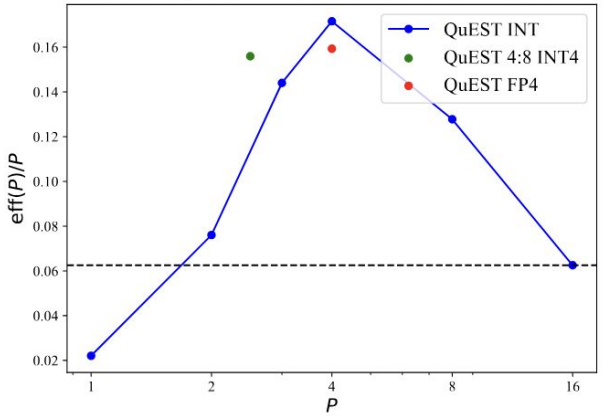


Figure 4. Illustration of the efficiency factors $\text{eff}(P)/P$, arising from our analysis, for different numerical precisions P and formats (INT, FP, INT+sparse). Higher is better. INT4 appears to have the highest efficiency among hardware-supported formats.

“Эффективный” размер модели при заданной битности

Оценка качества LLM

Удовлетворительного универсального протокола нет.
Надо подбирать специально под целевую задачу.

Перплексия

Wikitext 2

C4

0-shot / few-shot

Commonsense reasoning

Closed/Open-book QA

World Knowledge

Генеративные бенчмарки

MMLU-Pro

IFEval

BigBench

Длинный контекст

LongBench

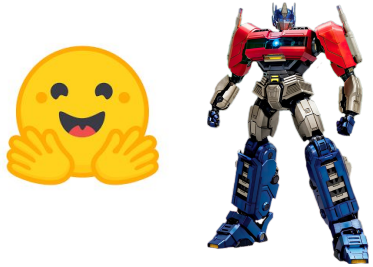
Needle in Haystack

Reasoning

GPQA

AIME/MATH 500

Интеграции в фреймворки



HuggingFace transformers поддерживает следующие форматы квантизации:

- Quanto (Round-To-Nearest)
- BitsAndBytes
- GPTQ
- AWQ
- VPTQ
- AQLM
- HIGGS и др.

https://huggingface.co/docs/transformers/en/main_classes/quantization

Интеграции в фреймворки

vLLM поддерживает следующие форматы квантизации:

- BitsAndBytes
- GPTQ
- AWQ
- AQLM
- HIGGS
- GGUF

https://docs.vllm.ai/en/latest/features/quantization/index.html

Интеграции в фреймворки

GGUF

GGUF включает в себя множество форматов скалярной и векторной квантизации в различной битности

Очень популярен для локального инференса

Квантизованную модель можно быстро приготовить

Не самый лучший по качеству

Популярный инференс движок

Поддерживает GGUF

https://huggingface.co/docs/hub/en/gguf

https://ollama.com/

Общие рекомендации

GPTQ
AWQ

Векторные квантизации:
AQLM
QuIP#
VPTQ
с дообучением при наличии ресурсов

Спасибо за