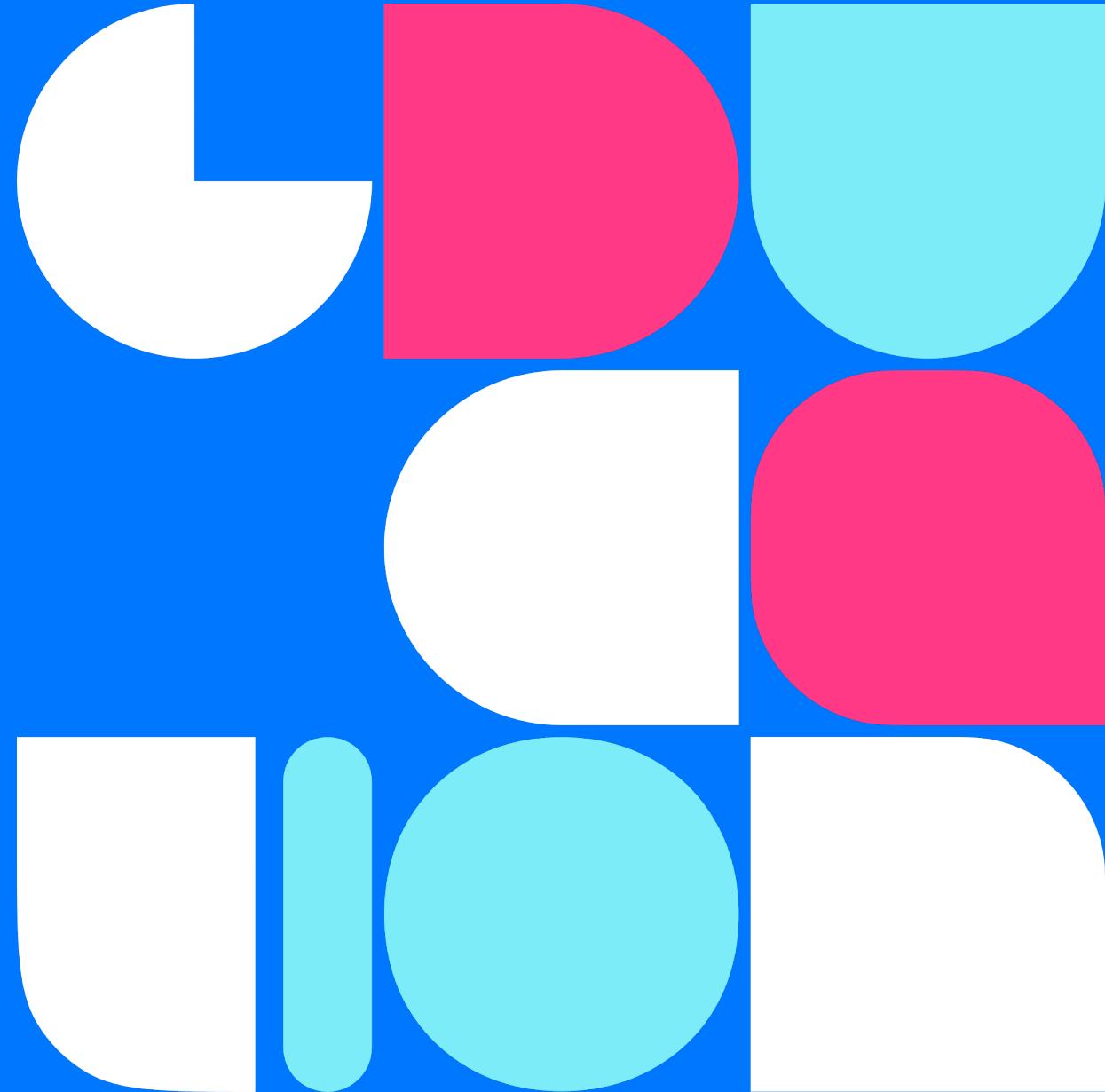




Transformers and beyond

Как развивались трансформеры

Егор Спирин



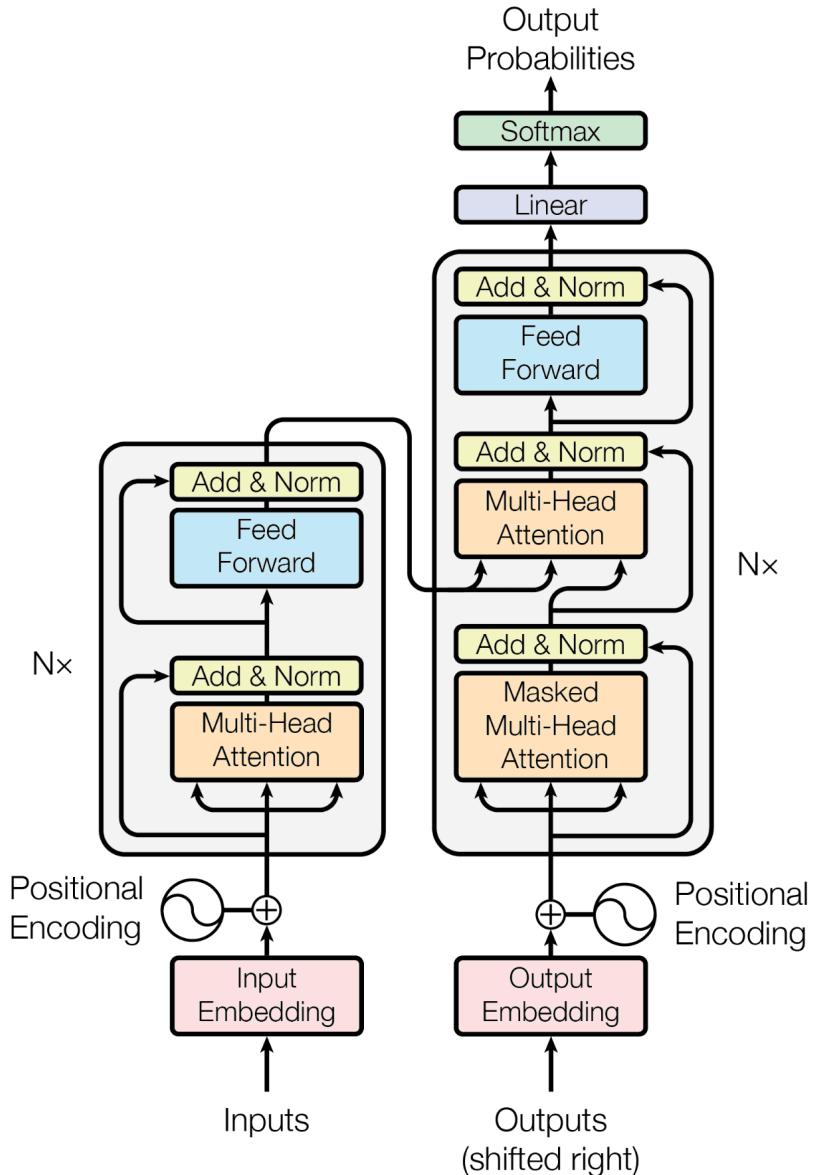
Quick recap



Here's for the crazy ones
The misfits. The rebels.
The troublemakers. The
round pegs in the
square holes.
The ones who see
things differently.
They're not afraid
of the status quo.
And they
change the world.

Attention is all you need

- 👉 **Transformer** — новая архитектура для естественного языка, проверка на машинном переводе
- 👉 Заточена под масштабирование на большие объемы данных — уходим от рекуррентной обработки данных
- 👉 Состоит из двух частей — Encoder и Decoder



Transformer Encoder

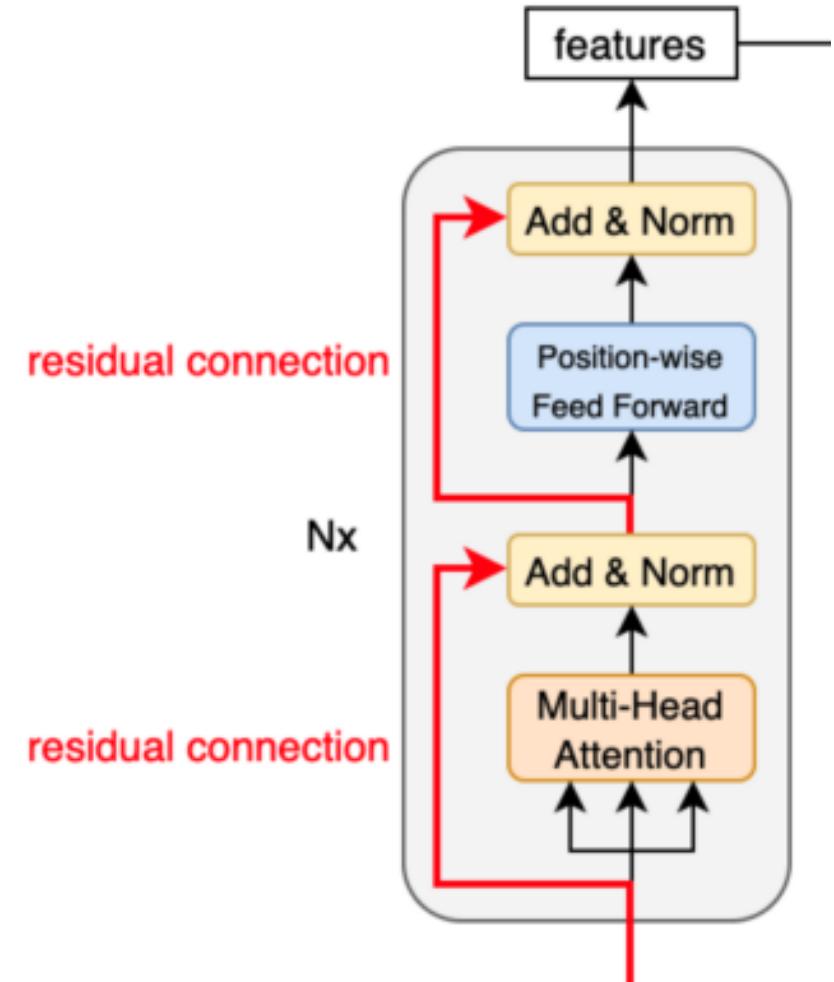
Transformer Encoder — состоит из последовательных
Transformer Encoder Layers

Transformer Encoder Layer:

- 👉 Self-Attention — механизм внимания, “изучаем” связь между элементами последовательности
- 👉 Feed-Forward — “запоминаем” признаки
- 👉 Слои нормализации и residual connection

Inputs: [batch size; seq len; hidden dim]

Features: [batch size; seq len; hidden dim]



Scaled Dot-Product Attention

Механизм внимания, вдохновленный ассоциативным хранилищем

1. $\langle K; V \rangle$ — хранилище ключей и значений
2. Q — запросы поиска: для q найти похожие k и взять соответствующие v с весами "схожести"

В мире трансформеров:

👉 Self-attention — используем общий вход

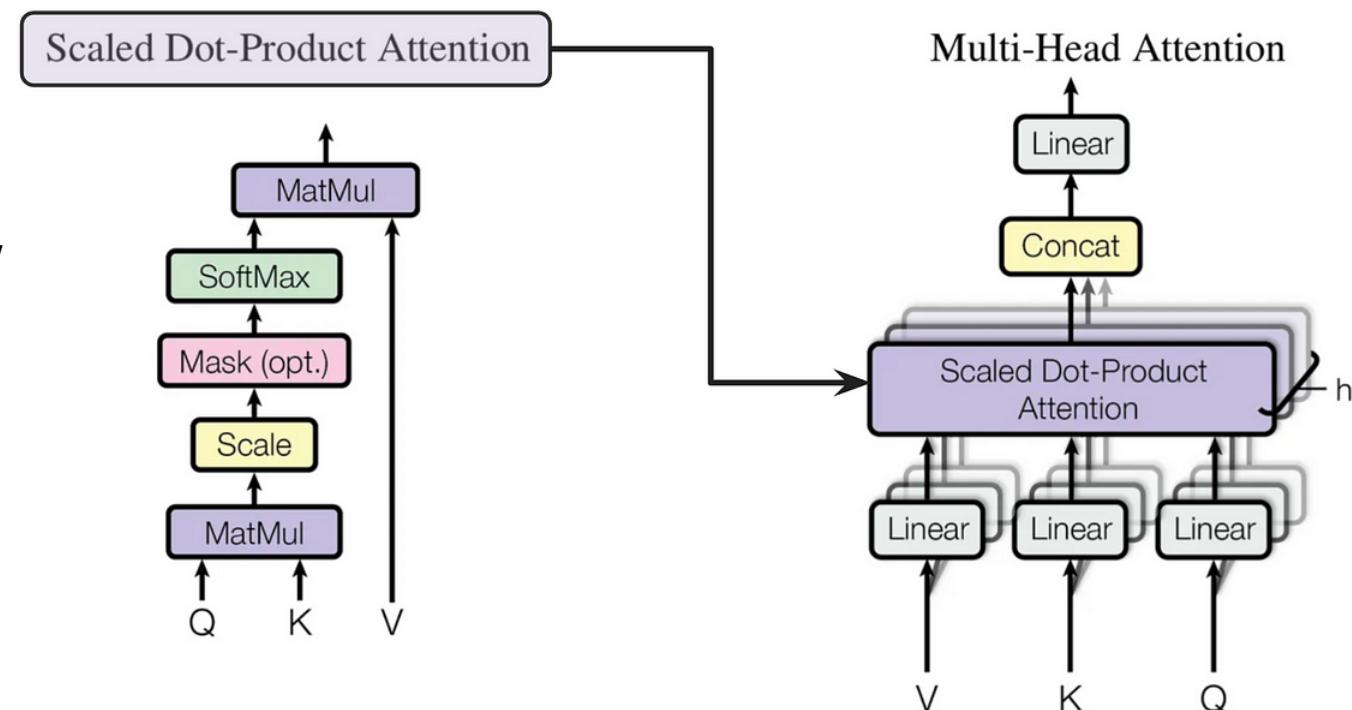
$$X \in [\text{seq_len}; \text{h_dim}]$$

$$Q = XW_q, K = XW_k, V = XW_v$$

👉 QK^T — attention map

$$(QK^T)_{i,j} — \text{"связь"} i \text{ и } j \text{ токена}$$

👉 Multi-Head Attention — несколько SDPA, но на меньшей размерности



Transformer Decoder

Transformer Decoder — состоит из последовательных **Transformer Decoder Layers**

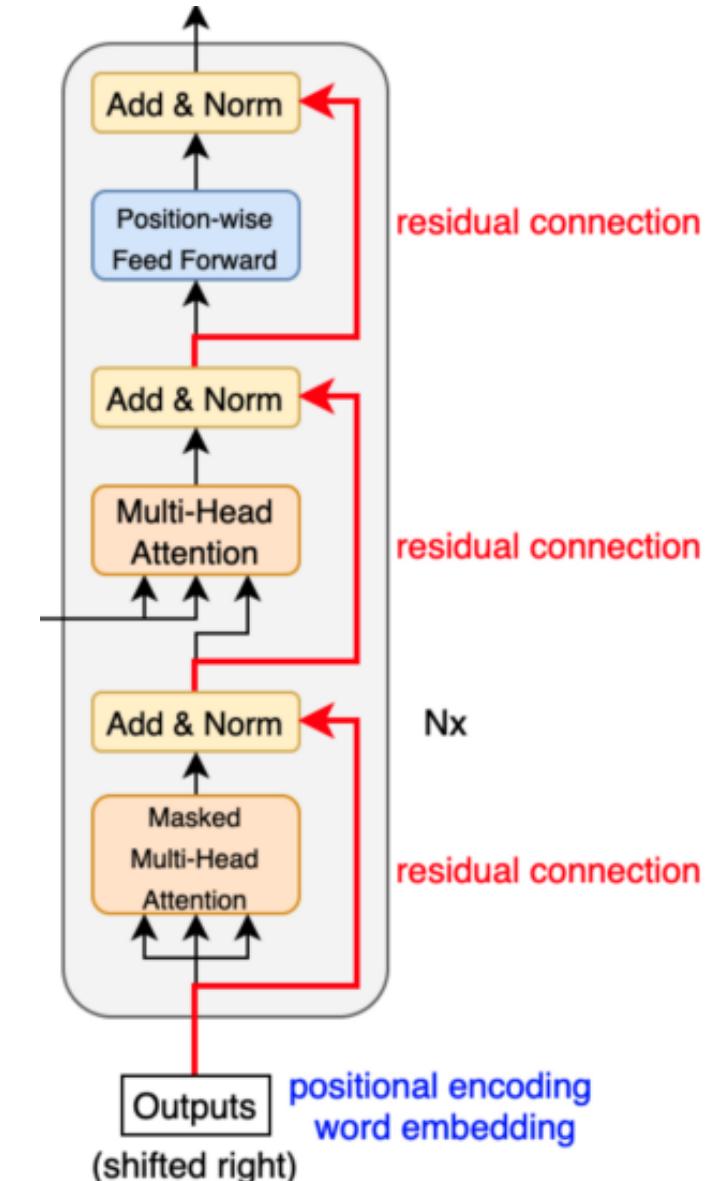
Transformer Decoder Layer:

- 👉 Практически полностью повторяет Transformer Encoder Layer
- 👉 Между Self-Attention и Feed-Forward добавляется Cross-Attention
- 👉 В Self-Attention запрещаем токенам смотреть в будущее через зануление части QK^T — применяется каузальная маска

Inputs: [batch size; seq len; hidden dim]

Conditions: [batch size; condition len; hidden dim]

Outputs: [batch size; seq len; hidden dim]

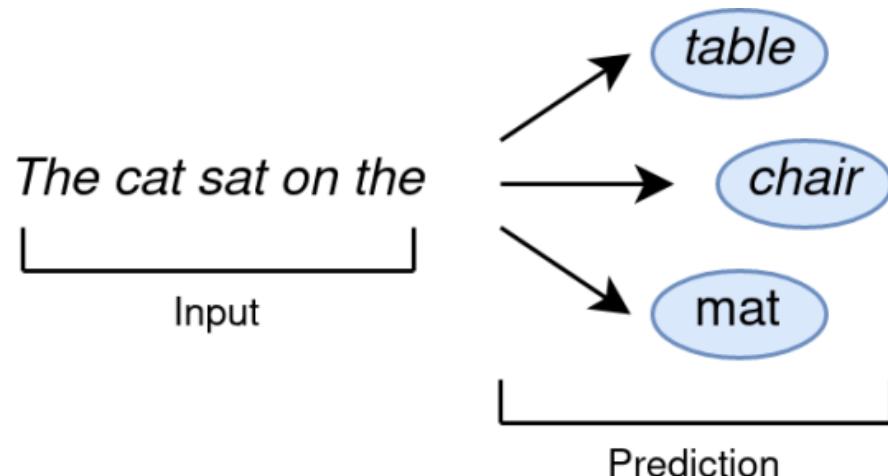


Языковое моделирование

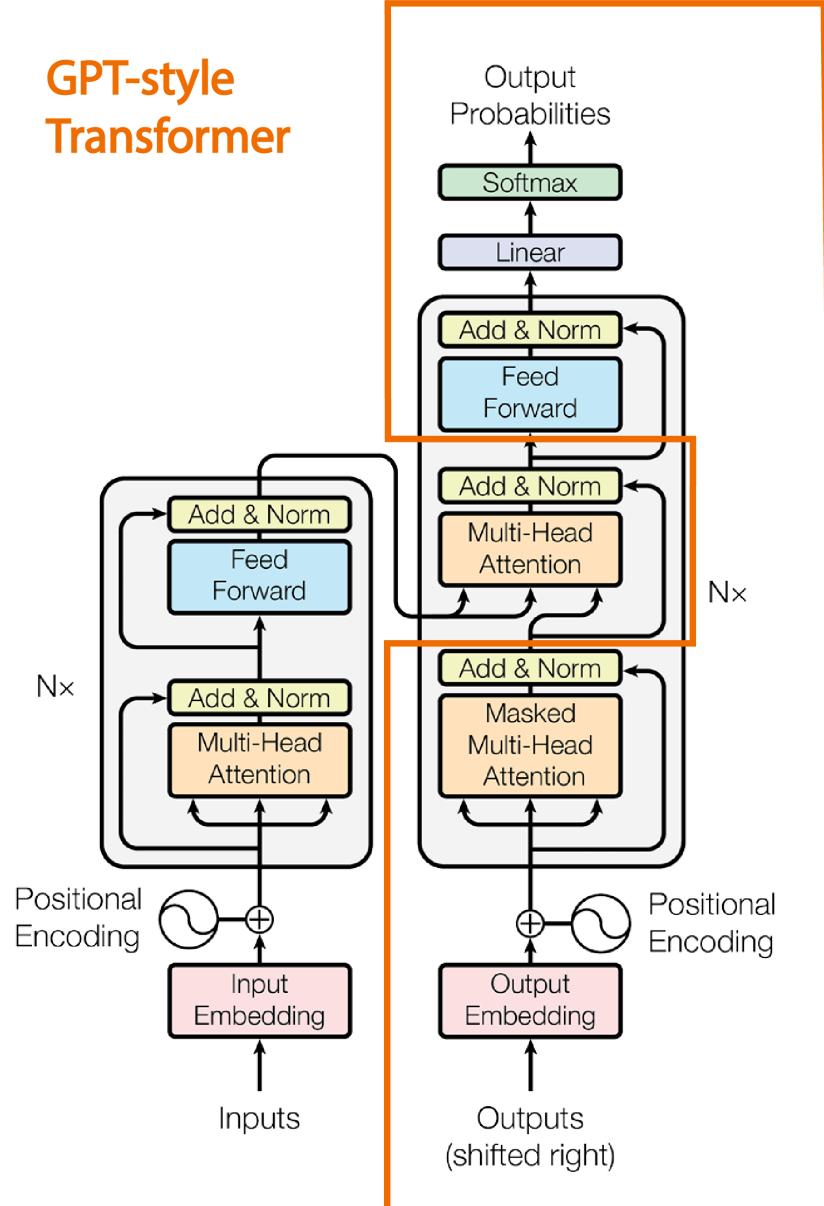
Языковая модель — авторегрессивная модель, предсказывает следующее слово в последовательности, основываясь на предыдущих словах, контексте

GPT and beyond:

- 👉 Transformer Decoder Layer, но без Cross-Attention
- 👉 Transformer Encoder Layer, но с каузальной маской и LM головой



GPT-style Transformer



Что было
далше

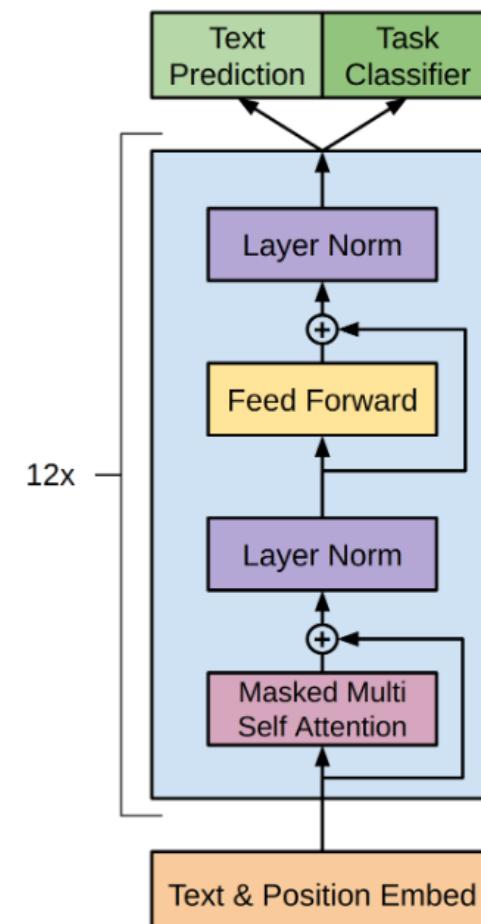


Transformers – не конкретная модель

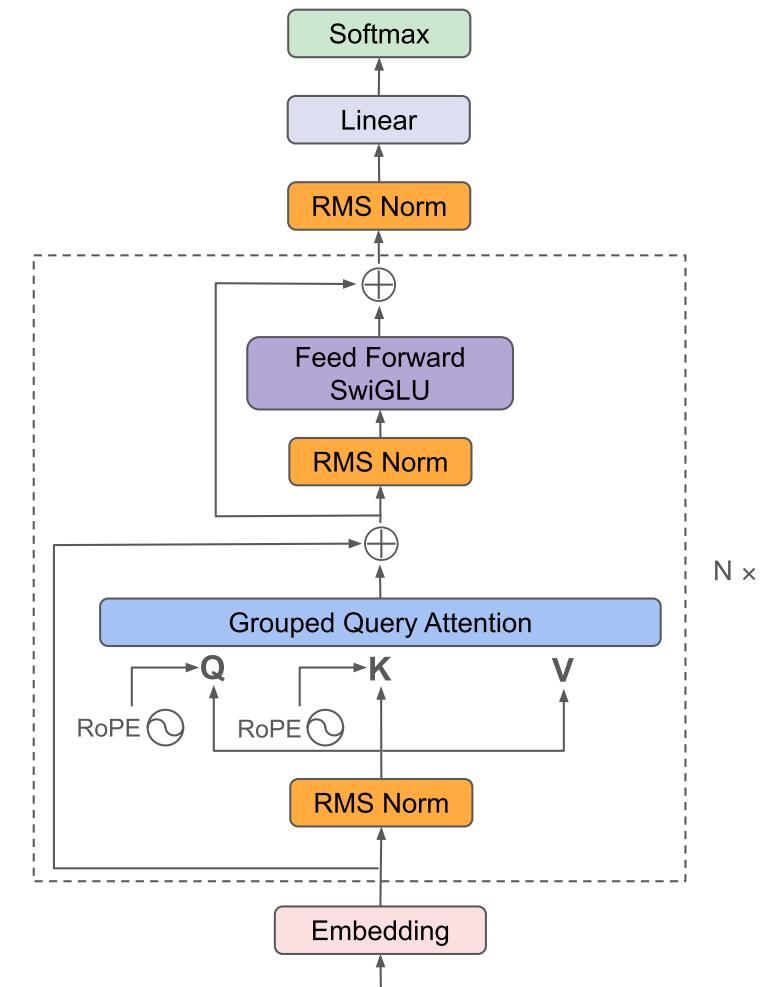
С 2017 модели сильно продвинулись вперед, они стали более стабильными и оптимизированными

Неизменным остались

- 👉 Механизм внимания для "замешивания" информации
- 👉 Блок нелинейности для "извлечения" и "хранения" информации



GPT

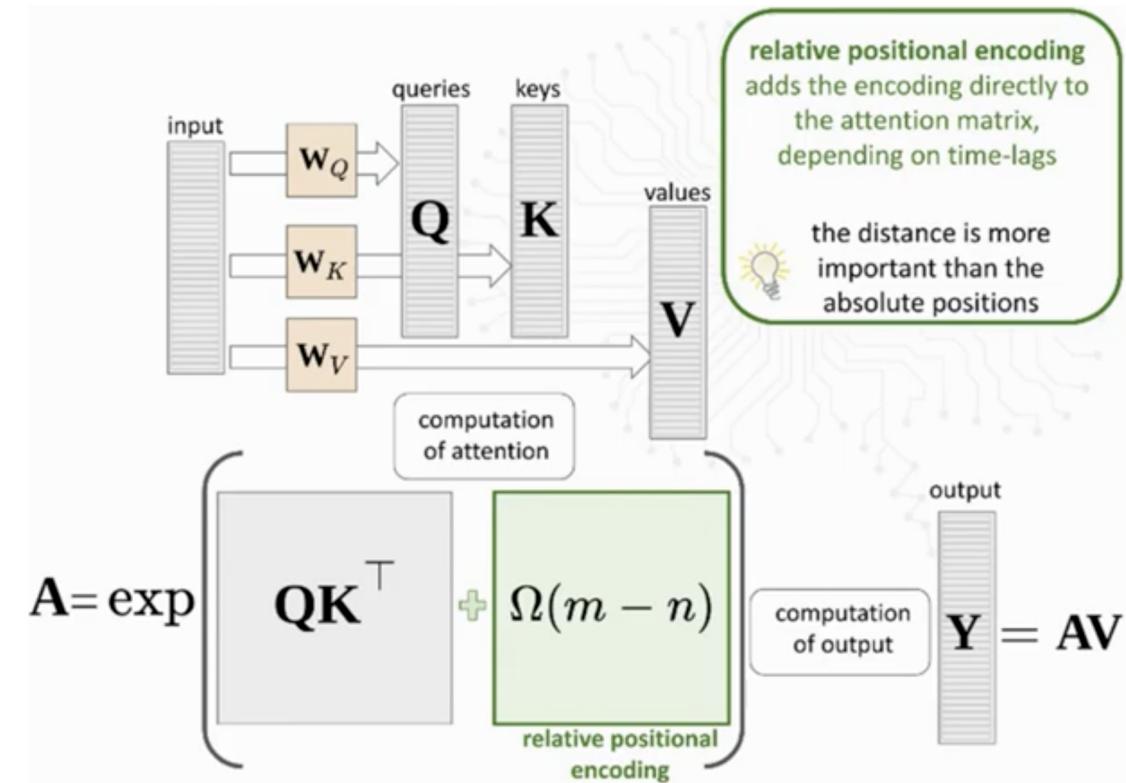


LLAMA-3

Positional embeddings

Механизм внимания инвариантен к позициям токенам — QK^T хранит семантические связи, не учитывает где эти токены расположены в последовательности

1. Абсолютные позиционные эмбеддинги — добавляем информацию о позиции в эмбеддинг токена, механизм внимания остается без изменения
2. Относительные позиционные эмбеддинги — добавляем информацию о разнице позиций между токенов, работаем с QK^T



Rotary Positional Embedding

$\langle q_m; k_n \rangle$ — обеспечивает передачу знаний между токенами на позициях m и n

Такое скалярное произведение может происходить где-то внутри $g(x_m, x_n, m - n)$

👉 Функция оперирует только токенами и расстоянием между ними

👉 Ванильный трансформер: $g(x_m, x_n, m - n) = \langle W_q x_m; W_k x_n \rangle$

Информация о $m - n$ не используется, позиции заложены в $x_{m,n}$

👉 Transformer-XL: $g(x_m, x_n, m - n) = \langle W_q x_m; W_k x_n \rangle + \langle W_q x_m; r_{m-n} \rangle + \langle u; W_k x_n \rangle + \langle v; r_{m-n} \rangle$

r_i — обучаемые эмбеддинги под относительные расстояния, u и v — обучаемые вектора

💡 **RoPE:** хотим найти такие $f_q(x_m, m)$ и $f_k(x_n, n)$, чтобы

$$\langle f_q(x_m, m); f_k(x_n, n) \rangle = g(x_m, x_n, m - n)$$

В итоге — считаем абсолютные, но кодируются относительно!

RoPE: 2D case

Пусть $x \in \mathbb{R}^2$ и определим $f_{\{q,k\}}(x_m, m) = (W_{\{q,k\}}x_m) \cdot e^{im\theta}$ — поворот вектора на угол $m\theta$

Такой поворот можно записать в матричной форме

$$f_{\{q,k\}}(x_m, m) = R_{m\theta} W_{\{q,k\}} \begin{pmatrix} x_m^{(1)} \\ x_m^{(2)} \end{pmatrix}, \text{ где } R_{m\theta} = \begin{pmatrix} \cos m\theta & -\sin m\theta \\ \sin m\theta & \cos m\theta \end{pmatrix}$$

В таком случае $g(x_m, x_n, m - n) = \langle f_q(x_m, m); f_k(x_n, n) \rangle =$

$$= \text{много математики} = (W_q x_m) \cdot (W_k x_n)^T \cdot e^{i(m-n)\theta}$$

Итого:

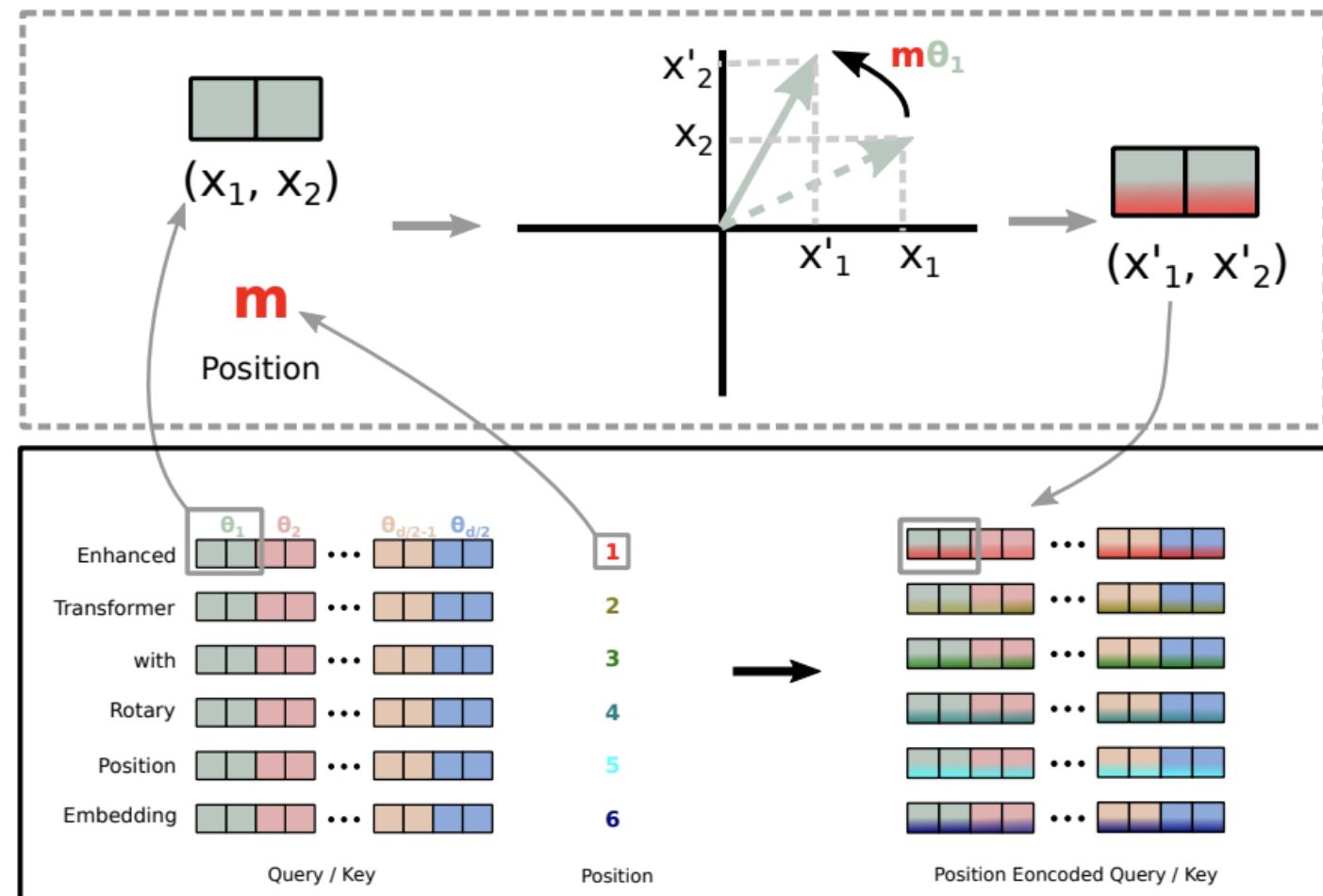
1. Считаем query и key
2. “Поворачиваем” query и key на углы m и n
3. Считаем скалярное произведение

RoPE: общий случай

Для $x \in \mathbb{R}^d$ — делим на $\frac{d}{2}$ частей и применяем 2D подход с $\theta_1, \dots, \theta_{d/2}$

$\Theta = \{\theta_i = 10000^{-2(i-1)/d}\}$ — совсем как в оригинальном позиционном кодировании

👉 На инференсе потенциально можем растягивать на новые длины



RoPE: на практике

Инициализируем тензор константных Θ , никаких обучаемых параметров

- 👉 Считаем Q и $K \rightarrow$ применяем RoPE — поворачиваем матрицы \rightarrow считаем МНА
- 👉 $R_{\Theta,m}^d \in \mathbb{R}^{d \times d}$ — блочно-диагональная матрица из $R_{m\theta_i}$, можно эффективно считать
- 👉 Работает с любой реализацией МНА: `flash-attn`, `torch.sdp`, ...
- 👉 Работает и со многими видами линейного механизма внимания

$$R_{\Theta,m}^d \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_{d-1} \\ x_d \end{pmatrix} \otimes \begin{pmatrix} \cos m\theta_1 \\ \cos m\theta_1 \\ \cos m\theta_2 \\ \cos m\theta_2 \\ \vdots \\ \cos m\theta_{d/2} \\ \cos m\theta_{d/2} \end{pmatrix} + \begin{pmatrix} -x_2 \\ x_1 \\ -x_4 \\ x_3 \\ \vdots \\ -x_d \\ x_{d-1} \end{pmatrix} \otimes \begin{pmatrix} \sin m\theta_1 \\ \sin m\theta_1 \\ \sin m\theta_2 \\ \sin m\theta_2 \\ \vdots \\ \sin m\theta_{d/2} \\ \sin m\theta_{d/2} \end{pmatrix}$$

GPT-J, GPT-NeoX, XPos, ... — доработали идею RoPE до современного вида

ALiBi

Делаем позиционное кодирование с возможностью увеличить длину контекста на инференсе

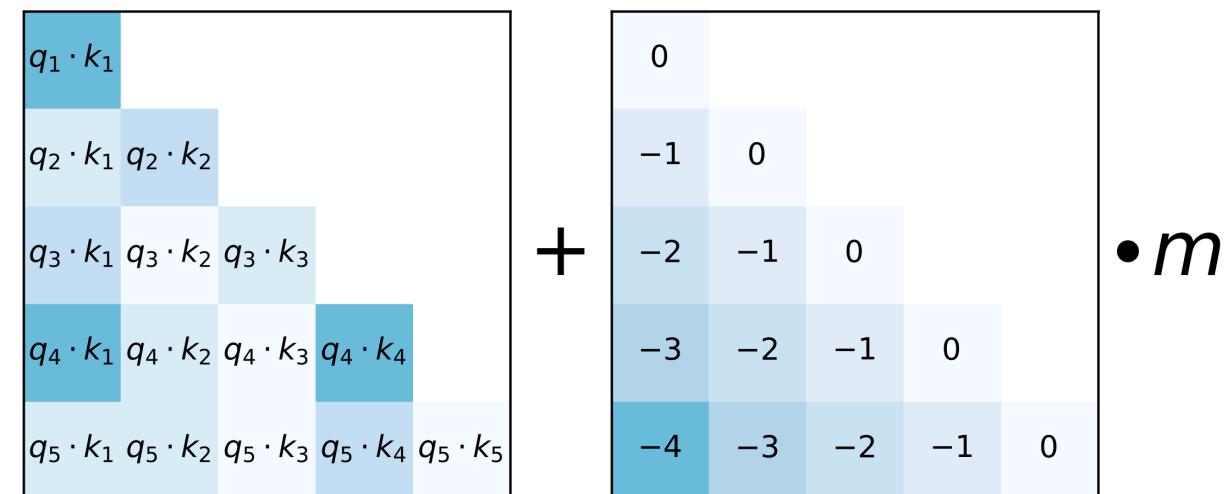
💡 **ALiBi (Attention with Linear Bias)** — добавляем статичный не-обучаемый баес

m — head-specific константа

👉 Для 8 голов: $1/2^1, \dots, 1/2^8$

👉 Для 16 голов интерполируем значения
 $1/2^{0.5}, 1/2^1, \dots, 1/2^{7.5}, 1/2^8$

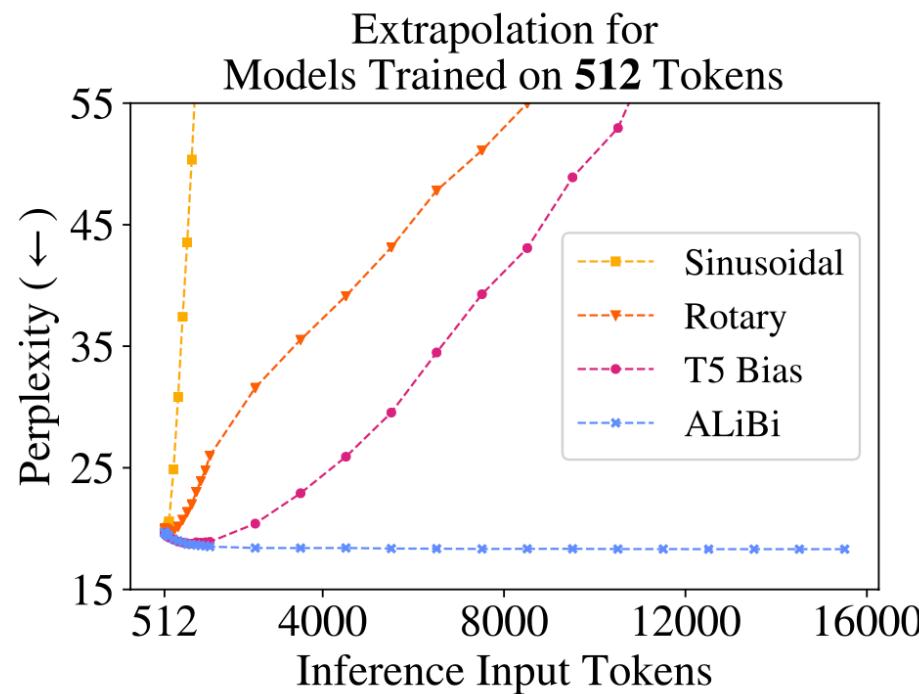
👉 Для n голов начинаем с $2^{-8/n}$ и
двигаемся с таким же шагом



Что выбрать?

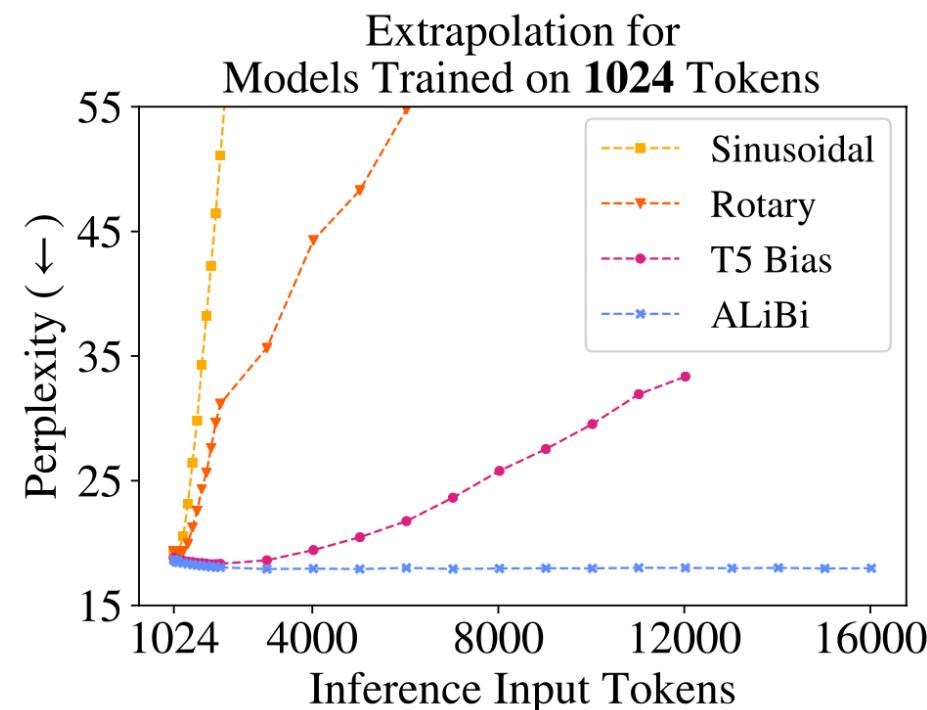
RoPE

- 👉 Интегрируется в любую реализацию МНА
- 👉 LLaMa, Gemma, Phi, Nemotron, ...



ALiBi

- 👉 Требуют внедрения в МНА, flash-attention поддержал только в 2.4 (Dec'23)
- 👉 Легче использовать на новых длинах
- 👉 Bloom, MPT, ...

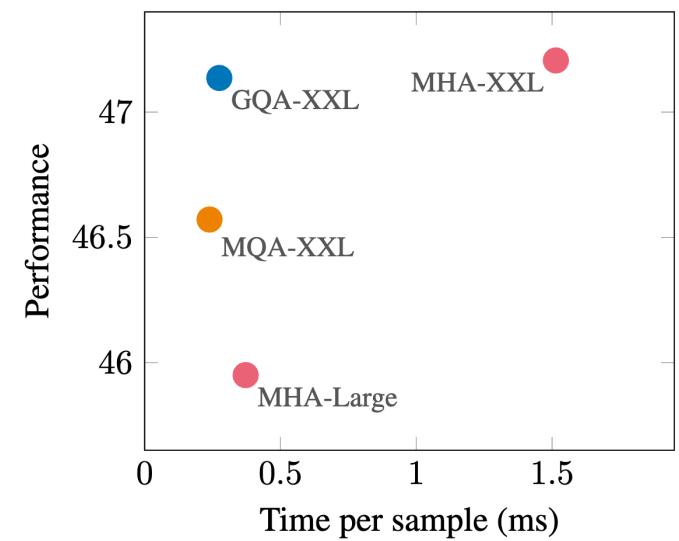
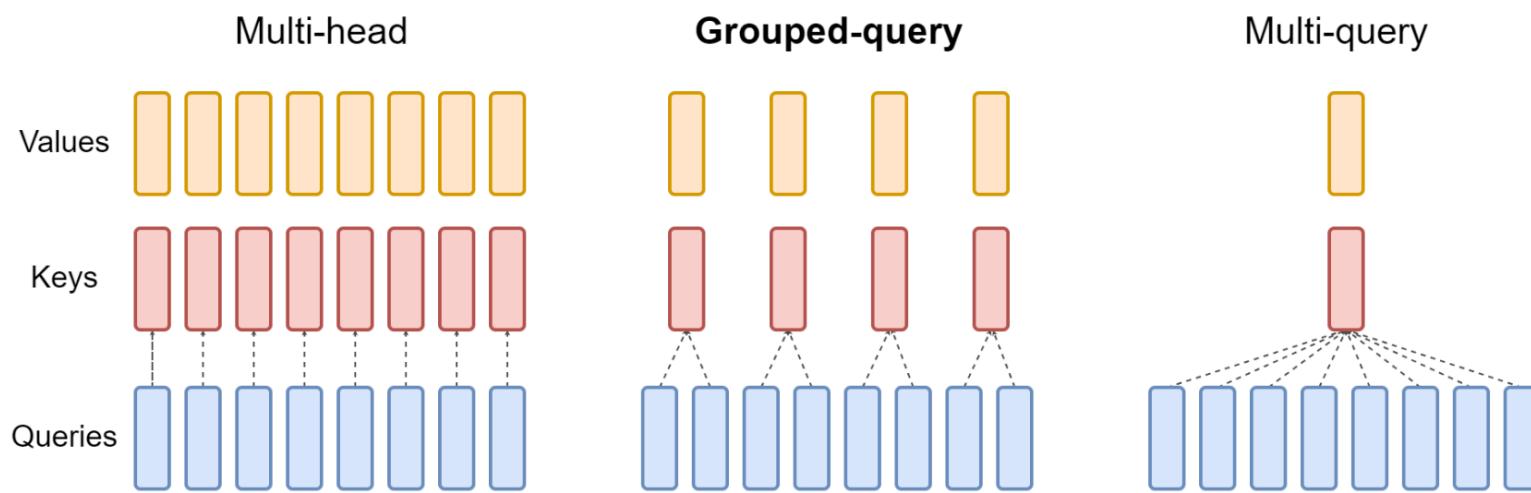


Grouped Query Attention

Чем длиннее контекст, тем дольше считать QK^T ... При этом помним (1) Q — отвечает за новые токены, а (2) K — отвечает за токены последовательности!

Multi-Query Attention — 1 голова K и V на все головы Q , значительно выигрываем в скорости

Grouped Query Attention — промежуточный вариант, баланс качества и скорости



[5] — Fast Transformer Decoding: One Write-Head is All You Need, Noam Shazeer, 2019

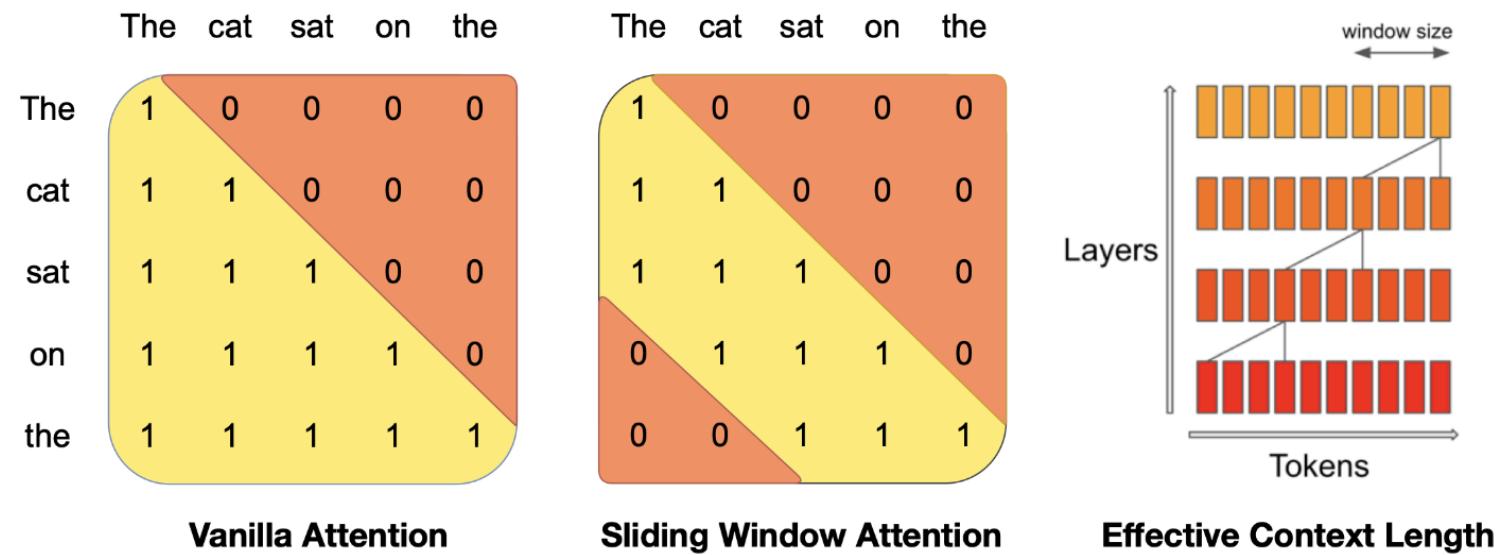
[6] — GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints, J. Ainslie, J. Lee-Thorp, M. De Jong et al., EMNLP'23

Sliding Window Attention

💡 Ограничиваем seq_len в рамках одного слоя внимания.

SWA, Dilated SWA, Global + SWA, ... — разные способы брать токены из последовательности

- 👉 Для окна W и k слоев, эффективный контекст $W \cdot k$
- 👉 Mistral-7B: для контекста в 16К и окна 4096 ускорение работы в 2 раза



Global & Local Attention

На практике, можем чередовать

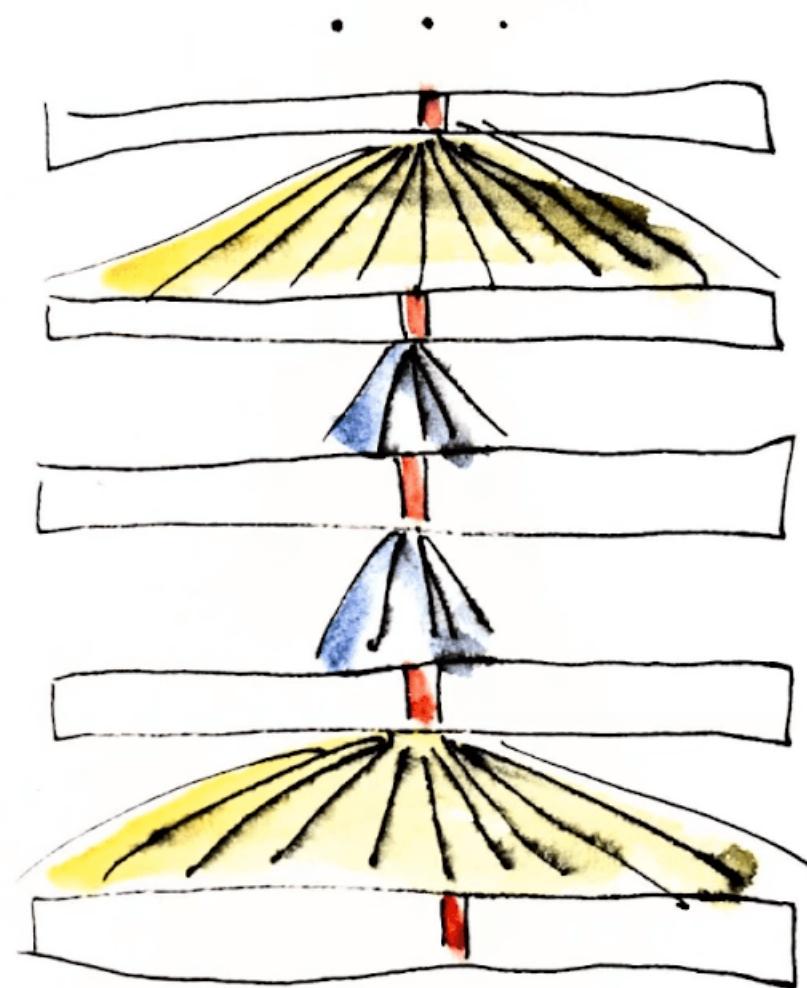
👉 **Global Attention** — смотрим на все доступные токены

👉 **Local Attention** — используем SWA

Возможный бенефит — можно менять размер окна на инференсе

sliding window	4096	2048	1024
perplexity (val. set)	1.63	1.63	1.64

Table 10 | Impact of changing the sliding window size at inference time for the 9B model.



[9] — Gemma 2: Improving Open Language Models at a Practical Size, Gemma Team, 2024

[10] — Smarter, Better, Faster, Longer: A Modern Bidirectional Encoder for Fast, Memory Efficient, and Long Context Finetuning and Inference, Benjamin Warner, Antoine Chaffin, Benjamin Clavie, 2024

Multi-Head Latent Attention

K и V — знания об обработанной части последовательности \Rightarrow
на инференсе можно переиспользовать при генерации следующего токена — **KV-cache**

1. МНА будет хранить кеш из $2n_h d_h l$ элементов для каждого токена
 n_h — число голов, d_h — размерность головы, l — число слоев
2. MQA и GQA сокращает размер кеша на новое число голов
3. **Multi-Head Latent Attention** — выучиваем скрытое представление $d_c \ll n_h d_h$ для K и V

$c_t^{KV} = W^{DKV} h_t$ — $W^{DKV} \in \mathbb{R}^{d_c \times d}$, down-projection матрица

$k_t^C = W^{UK} c_t^{KV}$ — $W^{UK} \in \mathbb{R}^{d_h n_h \times d_c}$, up-projection матрица для ключей

$v_t^C = W^{UV} c_t^{KV}$ — $W^{UV} \in \mathbb{R}^{d_h n_h \times d_c}$, up-projection матрица для значений

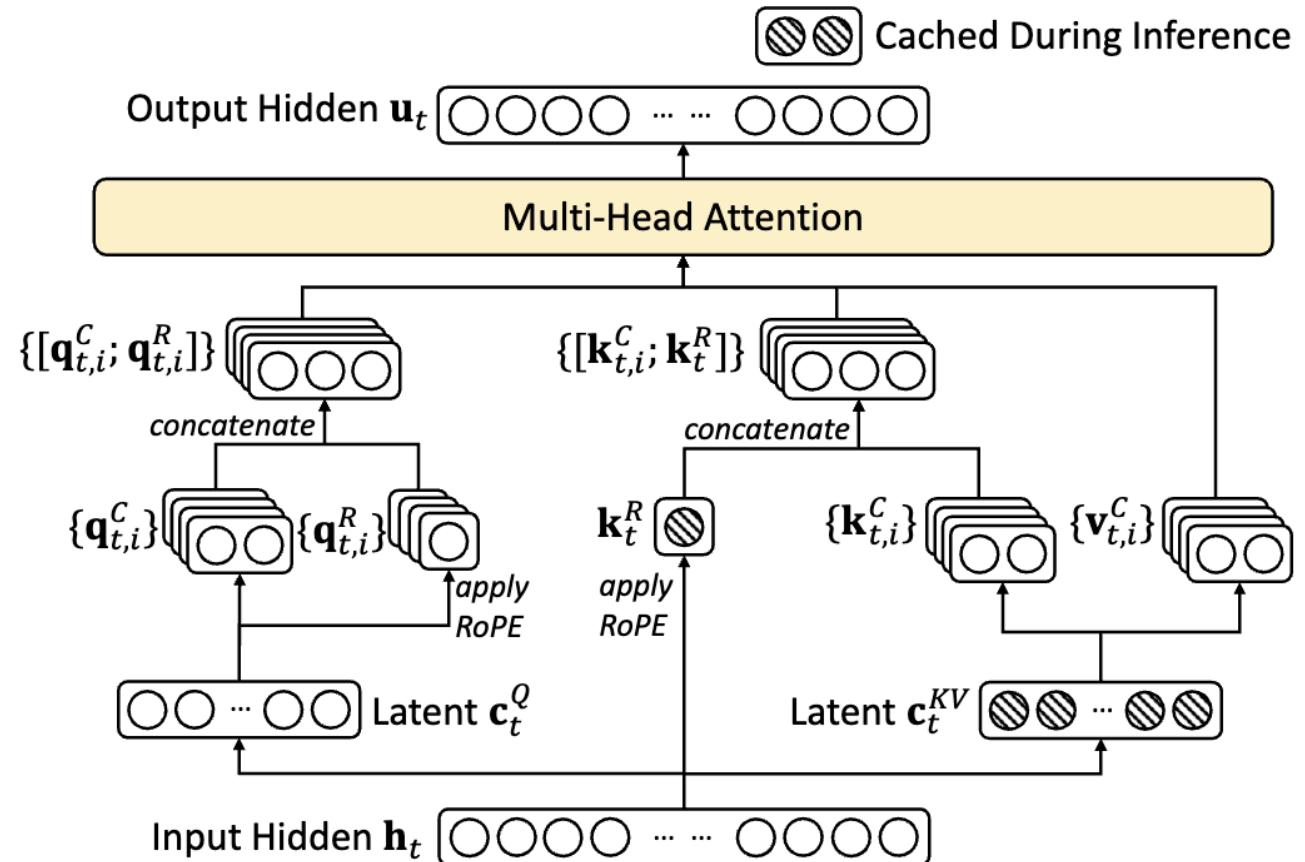
Для кеша достаточно хранить только $c_t^{KV} \Rightarrow$ его размер $d_c l$ элементов на каждый токен

При этом, $QK^T = XW_Q(C^{KV}W^{UK})^T = XW_q W^{UK^T} C^{KV^T} = XW_{\text{fused}} C^{KV^T}$ — нет доп. умножения

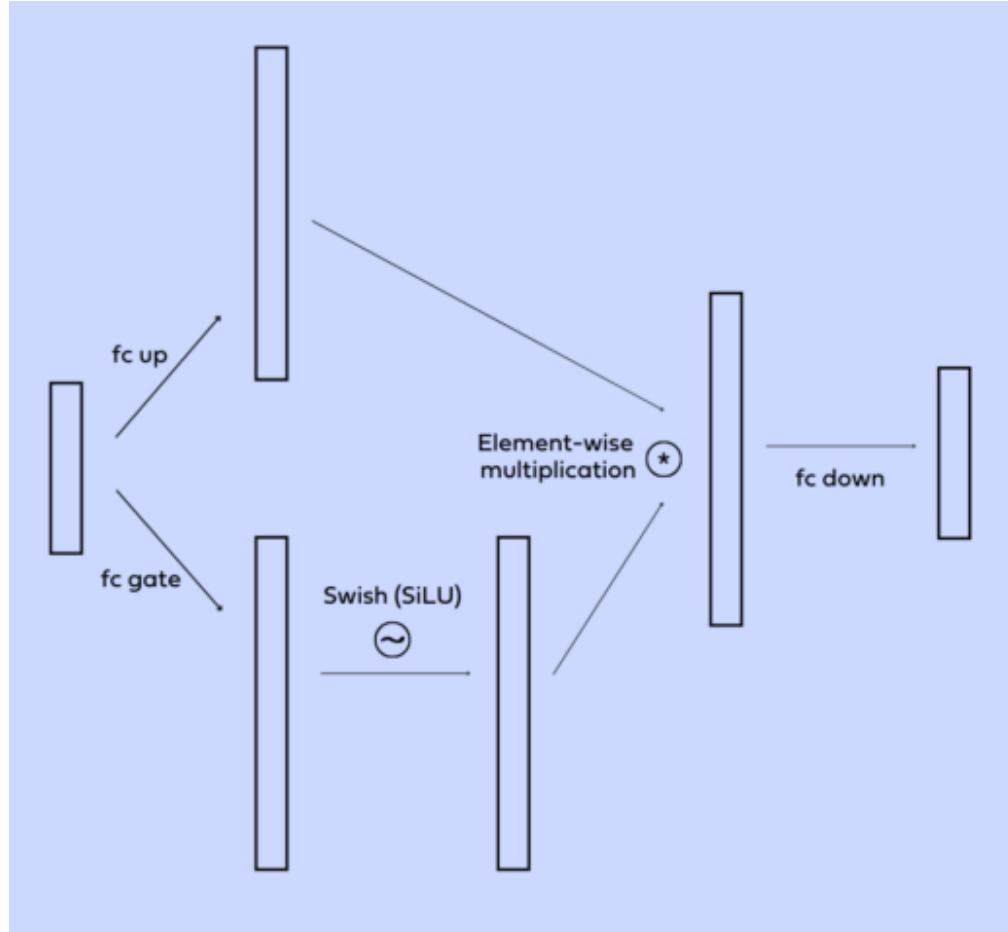
Multi-Head Latent Attention —



- 👉 Делаем аналогичный трюк для Q — уменьшаем memory activation во время обучения
- 👉 MHLA не дружит с RoPE — RoPE position-specific, нельзя применять к $W_{\text{fused}} = W_q W^{UK^T}$
- 👉 Делаем 50/50, часть голов без латента, но с RoPE
Для ключей RoPE-голова общая



SwiGLU



Стандартный FFN: $y = W_2(\text{Act}(W_1x + b_1)) + b_2$

👉 В качестве активации ReLU, GeLU, Swish, ...

👉 Как правило, $d_{\text{model}} \rightarrow 4 \cdot d_{\text{model}} \rightarrow d_{\text{model}}$

SwiGLU (Swish + Gated Linear Unit):

$$u = W_2x + b_2$$

$$g = \text{Swish}(W_1x + b_1)$$

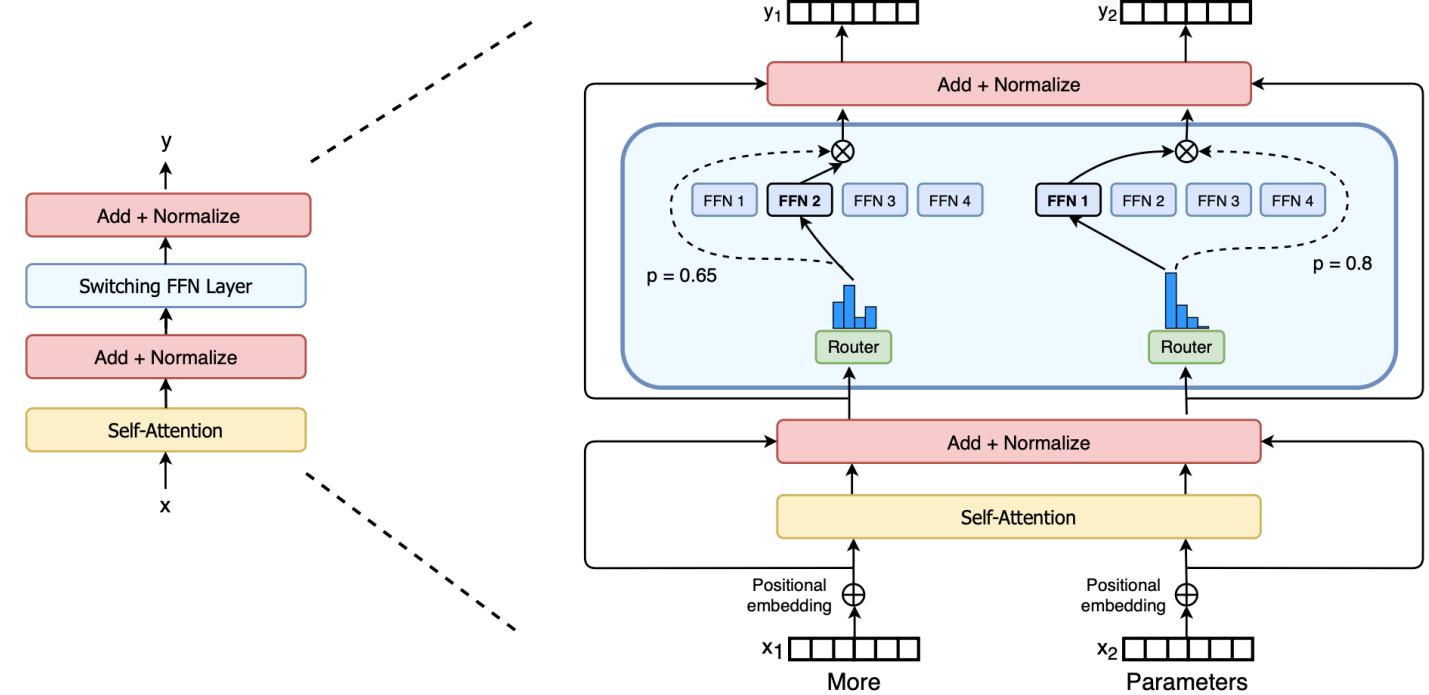
$$y = W_3(u \times g) + b_3$$

Изначально $8/3 \cdot d_{\text{model}}$ для баланса, но сейчас везде по-разному. Важно поддерживать кратность 64 или 128!

Mixture of Experts

MoE — sparse-слой, для каждого токена выбираются k из n экспертов. Позволяет эффективно наращивать число параметров — модель “знает” больше, но в моменте пользуется не всем

- 👉 Mixtral: 8 экспертов по 2 на каждый токен, из 47 млрд задействуется только 13
- 👉 Эксперт — любой FFN блок
- 👉 Роутинг — обучаемый слой для распределения токенов по экспертам



Собираем
модели



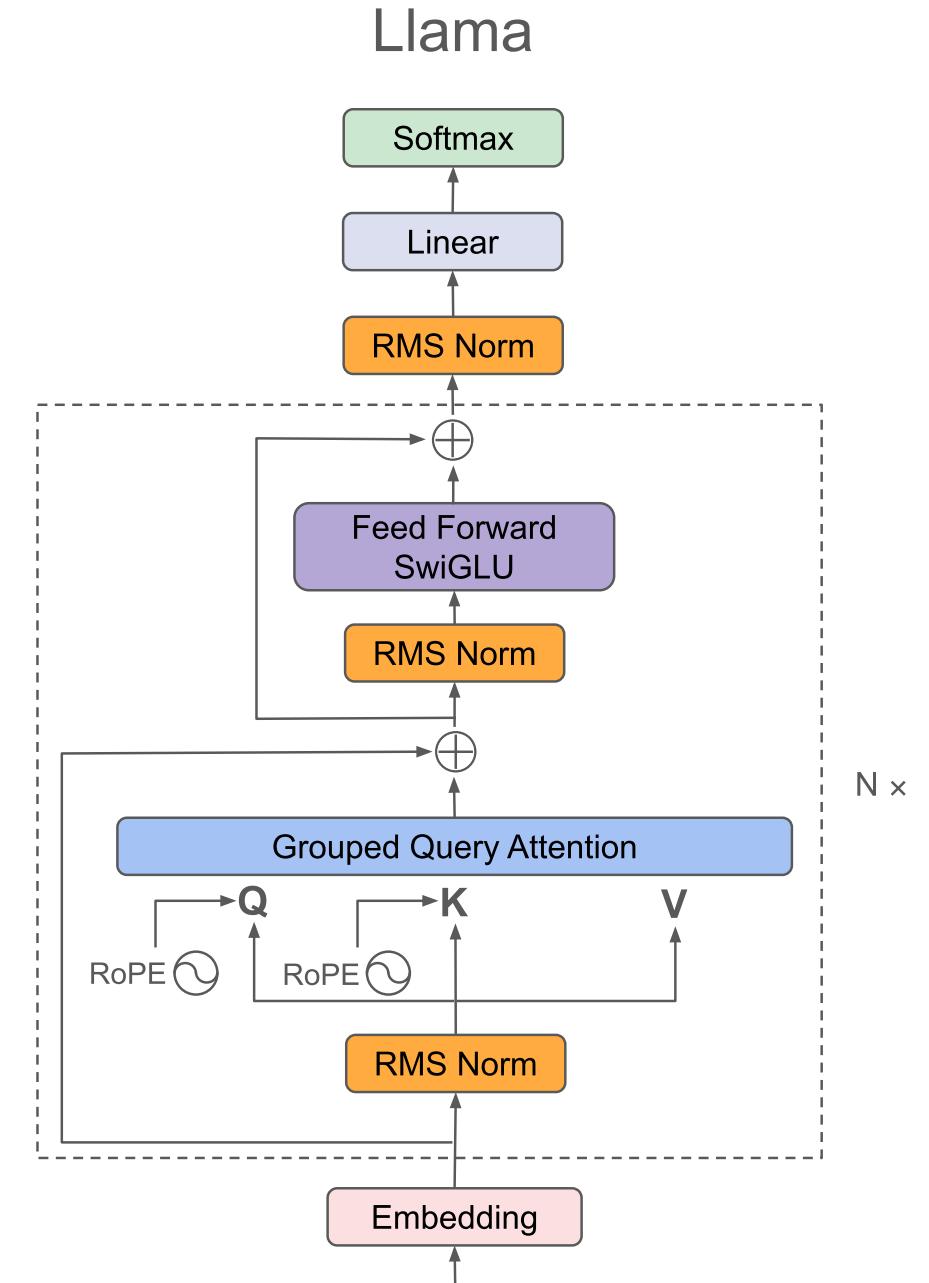
LLaMA-3

RMS Norm vs Layer Norm:

$$\bar{a}_i = \frac{a_i}{\text{RMS}(a)} g_i, \text{ где } \text{RMS}(a) = \sqrt{\frac{1}{n} \sum_{i=1}^n a_i^2}$$

Отказались от центрирования, но зато ускорили вычисления

- 👉 Линейные слои без использования bias
- 👉 Отсутствие dropout во время предобучения
- 👉 Fused-ядра для прямого и обратного прохода
- 👉 И еще много трюков для стабильного обучения
- 👉 Используем Pre- нормализацию вместо Post-



[15] — Root Mean Square Layer Normalization, Zhang et al., NeurIPS'19

[16] — Llama-3, Meta, 2024

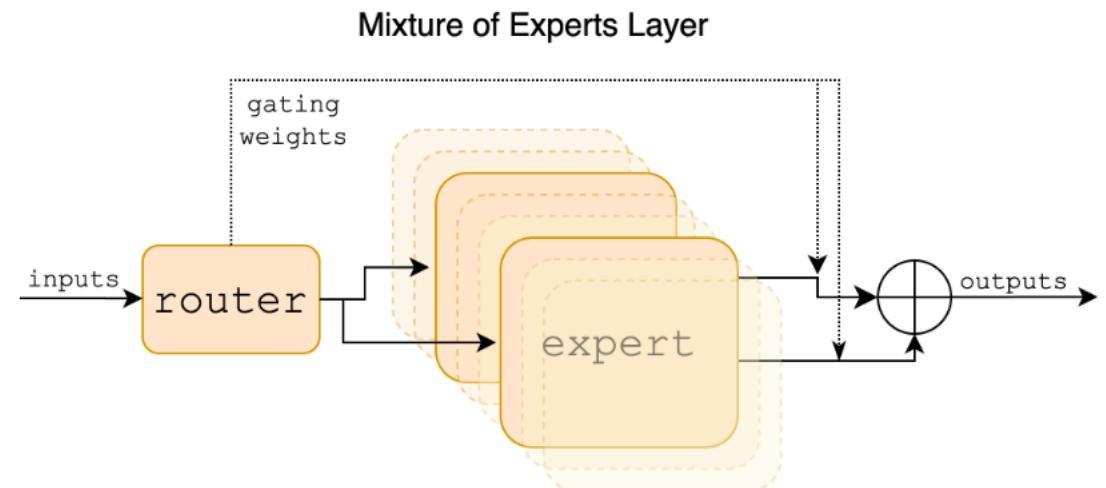
Mistral & Mixtral

Mistral 7B:

- 👉 GQA: 32 головы для query и по 8 для key и value
- 👉 SWA: 32 слоя и окно 4096
- 👉 RMSNorm, RoPE, SwiGLU

Mixtral 8x7B:

- 👉 MoE: 8 экспертов, каждый SwiGLU
- 👉 Для каждого токена выбирается 2 эксперта



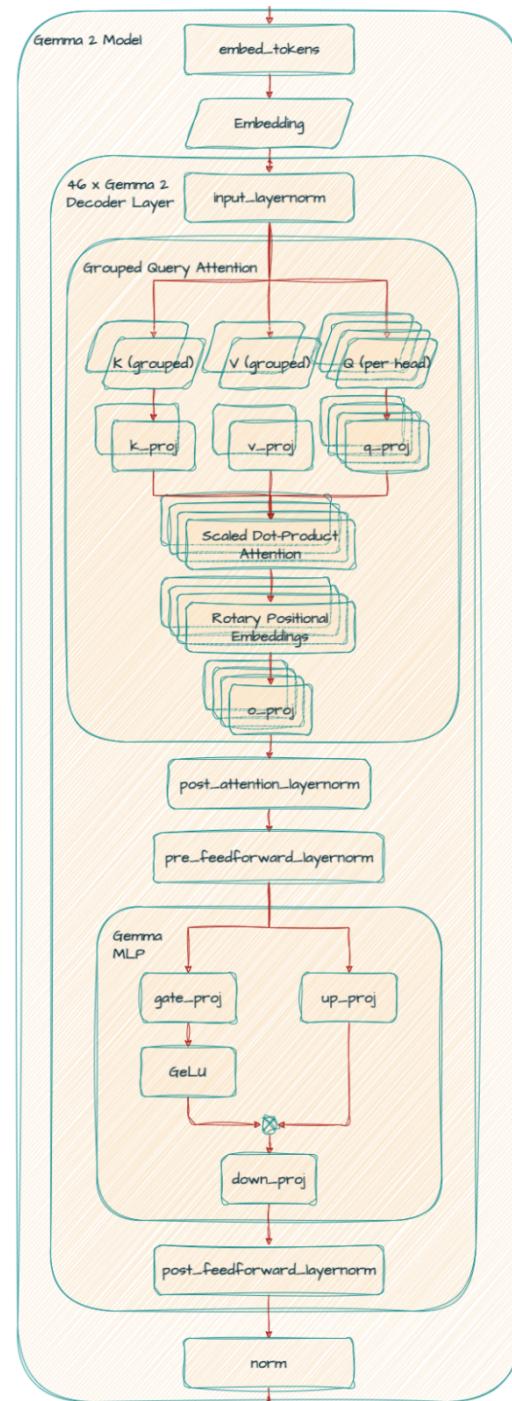
Gemma-2

- 👉 RMSNorm, SwiGLU, RoPE, GQA, ...
- 👉 И Pre-, и post- нормализация
- 👉 Local and Global Attention — SWA на каждом втором слое
- 👉 Logit Soft-Capping — запрещаем быть слишком уверенным в связи двух токенов и при предсказании следующего

Ограничиваем значения QK^T между $-\sigma$ и σ

$$QK^T \leftarrow \sigma \cdot \tanh\left(\frac{QK^T}{\sigma}\right)$$

- 👉 Немного другая работа с half-precision — хаки для стабильности, формально не отличается



Gemma-2 | Ablation

1. GQA не портит качество
2. При прочих равных лучше делать более глубокую модель, а не широкую
3. Размер окна для SWA можно уменьшать на инференсе (обучение было на 4096)

	Wide	Deep
Average (4 bench.)	50.8	52.0

Table 9 | Wide versus deep 9B models. Performance on 4 benchmarks, higher is better.

	MHA	GQA
Average (4 bench.)	50.3	50.8

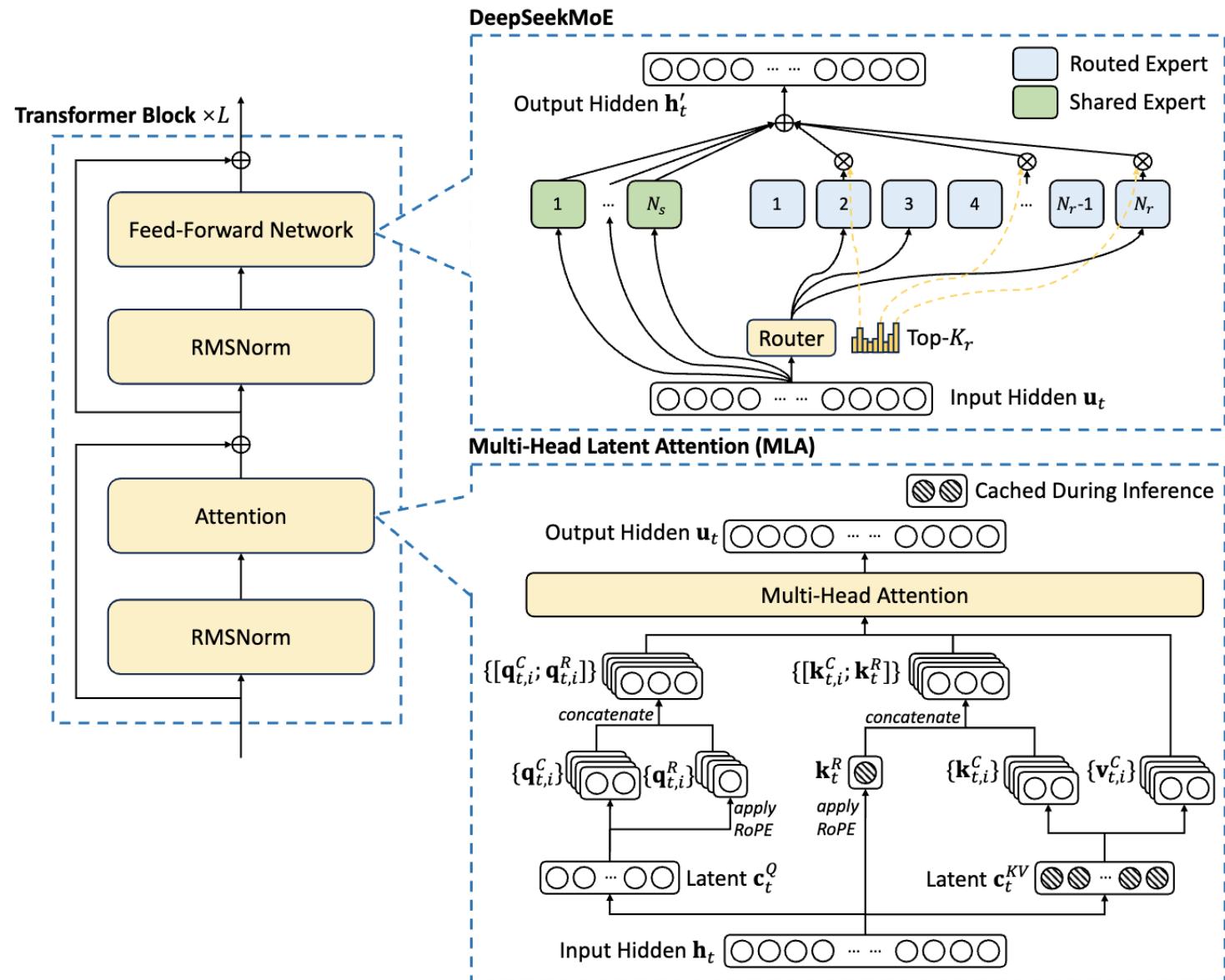
Table 8 | Comparing the impact of replacing Multi-Head Attention (MHA) with GQA on a 9B model averaged over 4 benchmarks.

	sliding window	4096	2048	1024
perplexity (val. set)	1.63	1.63	1.63	1.64

Table 10 | Impact of changing the sliding window size at inference time for the 9B model.

DeepSeek-v2/v3

- 👉 Multi-Head Latent Attention для механизма внимания
Скрытая размерность в 4 раза меньше исходной
- 👉 Mixture-of-Experts, каждый эксперт — SwiGLU
- 👉 В остальном все аналогично другим моделям



[11] — DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model, DeepSeek-AI, 2024

[19] — DeepSeek-V3 Technical Report, DeepSeek-AI, 2024

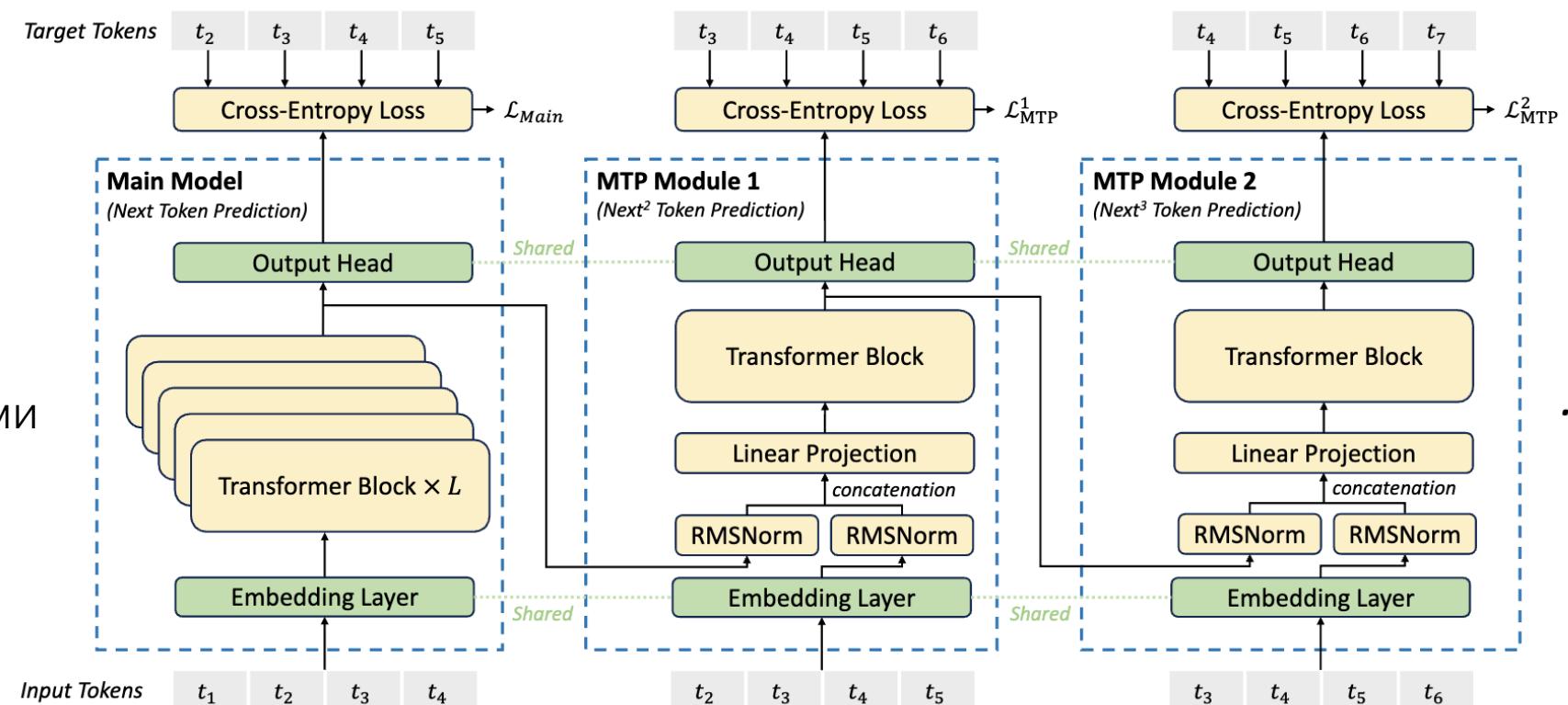
Multi-Token Prediction

💡 На каждом шаге предсказываем несколько токенов

МТР модуль — отдельный модуль, цепочка длины D (обучали с $D = 1$)

Каждый модуль считает стандартную Cross-Entropy, финальный лосс — сумма с обратно-пропорциональными коэффициентами

На инференсе МТР может использоваться для спекулятивного декодинга



Pre-training



Pre-Training

Стабильная архитектура + Много данных + Большой кластер = 

👉 Архитектуры обсудили:

- LLaMA — стандарт и база, точно будет работать
- Много экспериментальных хаков, постоянно выходят новые трюки
- Для больших моделей полезно переходить на MoE (а иногда и не для больших)

👉 Большой кластер:

- Инженерная работа — стек технологий и настройка самого кластера
- Инженерная работа — качественная реализация стабильной архитектуры

? Много данных — сколько, откуда и как брать

GPT-2

"Most prior work trained language models on a single domain of text, such as news articles, Wikipedia, or fiction books. Our approach motivates building as large and diverse a dataset as possible in order to collect natural language demonstrations of tasks in as varied of domains and contexts as possible."

- 👉 Web-страницы, на которые ссылались с Reddit
- 👉 Парсинг HTML, дедупликация
- 👉 ~8 млн документов, 40 гб текста

За счет разнообразия данных модель получилась многозадачной

"I'm not the cleverest man in the world, but like they say in French: **Je ne suis pas un imbecile** [I'm not a fool].

In a now-deleted post from Aug. 16, Soheil Eid, Tory candidate in the riding of Joliette, wrote in French: "**Mentez mentez, il en restera toujours quelque chose**," which translates as, "**Lie lie and something will always remain.**"

"I hate the word '**perfume**'," Burr says. 'It's somewhat better in French: '**parfum**'.

If listened carefully at 29:55, a conversation can be heard between two guys in French: "**-Comment on fait pour aller de l'autre côté? -Quel autre côté?**", which means "**- How do you get to the other side? - What side?**".

If this sounds like a bit of a stretch, consider this question in French: **As-tu aller au cinéma?**, or **Did you go to the movies?**, which literally translates as Have-you to go to movies/theater?

"Brevet Sans Garantie Du Gouvernement", translated to English: "**Patented without government warranty**".

Table 1. Examples of naturally occurring demonstrations of English to French and French to English translation found throughout the WebText training set.

GPT-3

1. Датасеты сильно больше — фильтрация CommonCrawl, дампа всего интернета
2. Добавление high-quality частей — книги, википедия
3. С помощью весов калибуруем соотношение данных при обучении

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

Детали фильтрация:

- 👉 Логистическая регрессия поверх текстовых статистик
- 👉 Дедупликация на основе MinHashLSH

Open-Source

LLaMA-1 также использовала смесь дампа интернета и доступные качественные источники

- 👉 Мультиязычный корпус
- 👉 Использование кода и в целом инженерных данных
- 👉 Фильтрация через маленькую LM на основе перплексии

Сейчас множество открытых аналогов, которые повторяют и дополняют эти идеи: [RedPajama](#), [FineWeb](#), [DCLM](#), ...

Dataset	Sampling prop.	Epochs	Disk size
CommonCrawl	67.0%	1.10	3.3 TB
C4	15.0%	1.06	783 GB
Github	4.5%	0.64	328 GB
Wikipedia	4.5%	2.45	83 GB
Books	4.5%	2.23	85 GB
ArXiv	2.5%	1.06	92 GB
StackExchange	2.0%	1.03	78 GB

Table 1: **Pre-training data.** Data mixtures used for pre-training, for each subset we list the sampling proportion, number of epochs performed on the subset when training on 1.4T tokens, and disk size. The pre-training runs on 1T tokens have the same sampling proportion.

Phi Family

💡 Нужны не огромные корпуса текста, нужны качественные данные

1. Отфильтрованные по языку программирования The Stack и StackOverflow — фильтрация через разметку GPT-4 и обучение классификатора
2. Синтетический textbook датасет <1 млрд токенов сгенерированных GPT-3.5
3. Маленький синтетический датасет заданий на программирование — ~180M токенов питоновских задач и их решений

Model		Size	Training tokens	Score	HumanEval
CodeGen-Mono-350M	[NPH ⁺ 23]	350M	577B	19%	13%
CodeGen-Mono-16.1B	[NPH ⁺ 23]	16.1B	577B	38%	29%
Replit	[Rep23]	2.7B	525B	37%	22%
StarCoder	[LAZ ⁺ 23]	15.5B	1T	51%	34%
phi-1-base		1.3B	7B	37%	29%
phi-1-small		350M	7B	45%	45%
phi-1		1.3B	7B	52%	51%

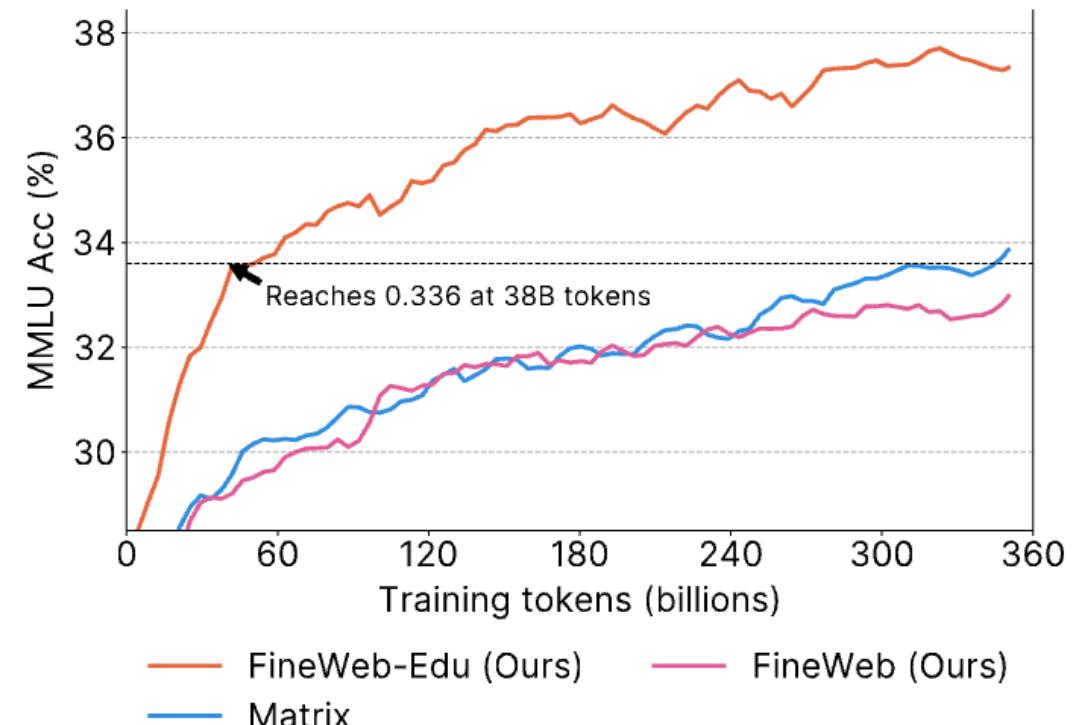
Table 2: LLM graded Understanding scores on 50 new unconventional coding problems.

FineWeb-Edu

💡 Пусть LLM сами фильтрует данные для обучения новых — наша задача сформулировать критерии фильтрации

FineWeb-Edu — оцениваем каждый документ по 5-балльной шкале на полезность/образовательность контент

1. Размечаем часть FineWeb через LLaMA-3
2. Обучаем легкий классификатор
3. Фильтруем весь FineWeb — удалили >90% датасета



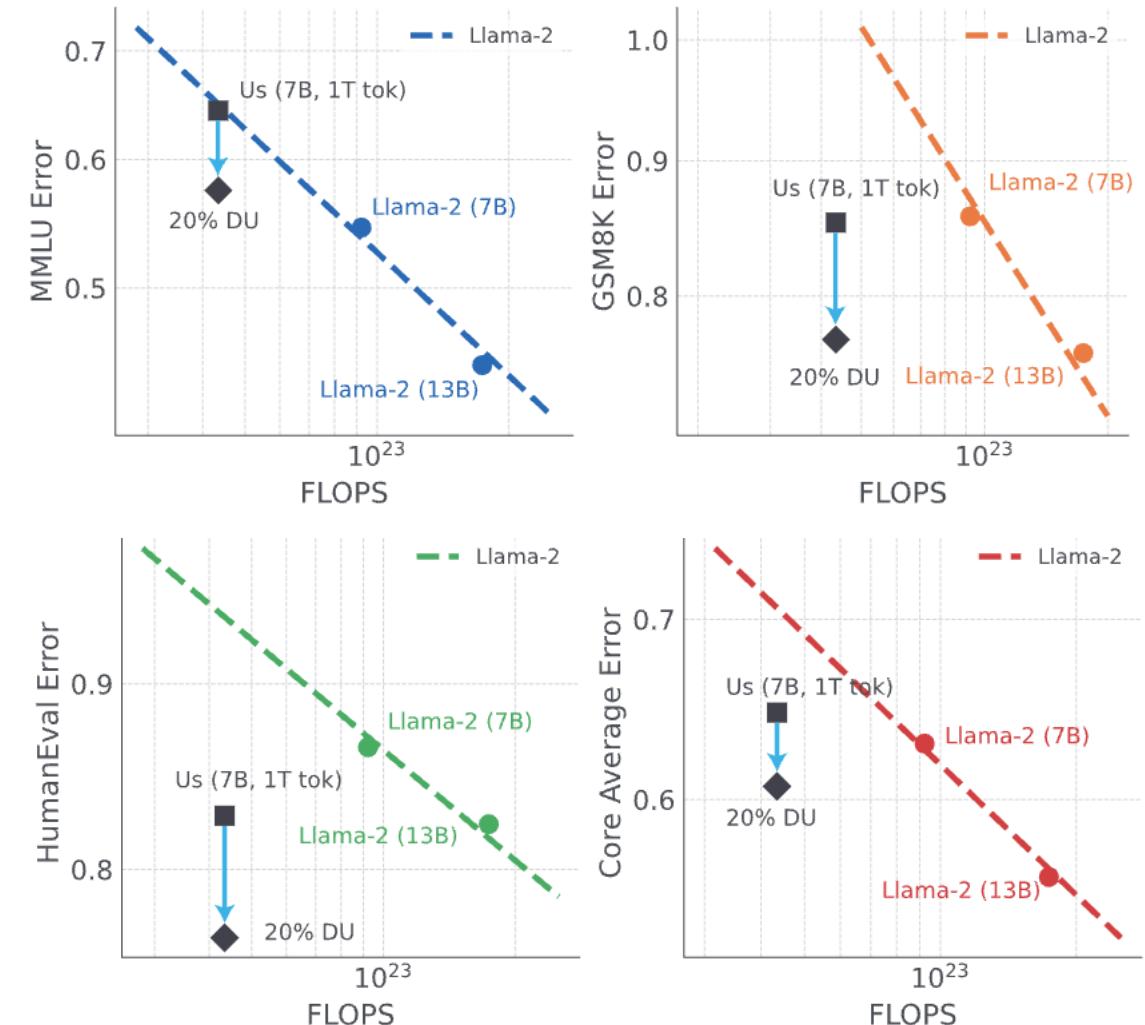
Domain Upsampling, Knowledge Annealing

Databricks Mosaic: "Upsamples domain specific datasets relative to Common Crawl at the end of training"

LLaMA-3: "During pre-training on the final 40M tokens ... we also adjusted the data mix to upsample data sources of very high quality"

Делим данные на категории:

- 👉 Large-Scale интернет, Small-Scale интернет, доменные данные, task-specific данные
- 👉 Ближе к концу добавляем все больше чистых данных
- 👉 Отдельно выделяем данные с кодом, которые всегда полезны, но также разной чистоты



А еще!

1. Дистилляция — маленькие LLM лучше обучать через дистилляцию больших
2. Объем предобучения меряется в числе токенов
 1. Наращиваем размер батча постепенно
 2. Увеличиваем длину контекста постепенно
3. Scaling Laws — работают, но со звездочкой
 1. Сильно зависит от архитектуры модели, ее реализации и кластера — учимся считать MFU
 2. На какой вопрос хотим ответить — сколько данных? Как долго? Какое будет качество?

	from scratch	distilled
Average (3 bench.)	60.3	67.7

Table 6 | Comparison between a 2B model trained over 500B tokens either from scratch or with distillation from a 7B model.

	200M	400M	1B
from scratch	23	19	17
distilled (7B)	21	17	15

Table 7 | Perplexity measured on a validation set of models of different sizes trained with or without distillation. The teacher has 7B parameters.

[Gemma-2 Tech Report](#)

На этом все



Егор Спирин — vk.com/boss
VK Lab — vk.com/lab