

Генеративные модели в NLP

Современные подходы к генерации текста и большие языковые модели

Начать лекцию →



О спикере

По науке:

- бак. ММФ НГУ
- маг. ФПМИ МФТИ feat Tinkoff
- асп. ММФ НГУ

По индустрии:

- из мира разработки
- ex-{ Promsoft, Tinkoff }
- Pretrain TechLead, GigaChat,
SberDevices



I. Введение и мотивация

Что такое генеративные модели в NLP?

Определение и основные принципы

Генеративная модель изучает распределение данных $P(x)$ для создания новых, похожих примеров

Дискриминативные модели

$P(y|x)$ - классификация

Генеративные модели

$P(x)$ - генерация

"Какой это класс?"

"Как создать похожий текст?"

- Sentiment analysis
- Named Entity Recognition
- Text classification

- Text generation
- Machine translation
- Dialogue systems

Фундаментальные вызовы в генерации текста

Вычислительная сложность

Проблемы

- Экспоненциальный рост словаря V^n
- Долгосрочные зависимости в тексте
- Семантическая связность
- Дискретность токенов vs непрерывная оптимизация

Задачи

- Генерация текста
- Машинный перевод
- Диалоговые системы
- Суммаризация
- Генерация кода

Автогрессивная формулировка

$$P(x_1, x_2, \dots, x_n) = \prod_{i=1}^n P(x_i | x_1, \dots, x_{i-1})$$

Предсказание следующего токена на основе предыдущих

II. Исторический контекст

Эволюция подходов к генерации текста

Три эпохи развития

Статистические методы (1990-2010)

- N-gram модели: $P(w_i | w_{i-n+1}, \dots, w_{i-1})$
- Скрытые марковские модели (HMM)
- Ограничения: короткая память, разреженность данных

Нейронные сети (2010-2017)

- RNN → LSTM → GRU → seq2seq
- Введение attention mechanism
- Ограничения: последовательная обработка, vanishing gradients

Transformer эра (2017-настоящее)

- "Attention is All You Need" → революция в архитектуре
- GPT/BERT paradigm → современные LLM
- Scaling laws → предсказуемые улучшения

Ключевые прорывы, изменившие область

Три революционных момента

 **Attention Mechanism**

 **Transfer Learning**

 **Scaling Laws**

2017

- Параллельная обработка
- Прямой доступ к любой позиции
- Основа для Transformer

2018-2019

- Предобучение + дообучение
- Эффективное использование данных
- Универсальные представления

2020+

- Предсказуемые улучшения
- Emergent abilities
- Планирование ресурсов

Результат: Современные LLM

Синергия этих трех прорывов создала основу для GPT-4, Claude, Gemini и других современных моделей

III. Архитектурные основы

Transformer Architecture: Фундаментальные принципы

Проблемы RNN/LSTM

- Последовательная обработка - нет параллелизации
- **Vanishing gradients** на длинных последовательностях
- **Ограниченнная память** даже в LSTM
- **Квадратичная времененная сложность**

Решение: Transformer

- **Параллельная обработка** всех позиций
- **Прямой доступ** к любой позиции
- **Масштабируемость** на длинные тексты
- **Эффективность** на GPU

Ключевая идея: Self-Attention

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V$$

Каждый токен может обращаться к любому другому токену напрямую

Multi-Head Attention и Position Encoding

МНА

```
MultiHead(Q,K,V) = Concat(head1, head2, ..., headn)@W^0  
где headi = Attention(QW^Q_i, KW^K_i, VW^V_i)
```

Преимущества:

- Множественные аспекты отношений между токенами
- Параллельная обработка разных представлений
- Композиционность сложных паттернов

Position Encoding

Transformer **position-agnostic** - нужно явно кодировать порядок (cos/sin после nn.Embedding)

Современные подходы: Rotary Position Embedding (RoPE), Alibi, xPos

Rotary Position Embedding (RoPE)

Проблемы классического Position Encoding

sin/cos embedding

- **Абсолютное кодирование** позиций
- **Фиксированная длина** последовательности
- **Слабая интерполяция** для новых позиций
- **Нет информации** об относительном
расстоянии

RoPE

- **Относительное кодирование** позиций
- **Экстраполяция** на большие длины
- **Сохранение внутреннего произведения**
- **Эффективная реализация**
- Используется в **LLaMA, Qwen, GigaChat**
- Масштабируется на длинные контексты

RoPE "на пальцах"

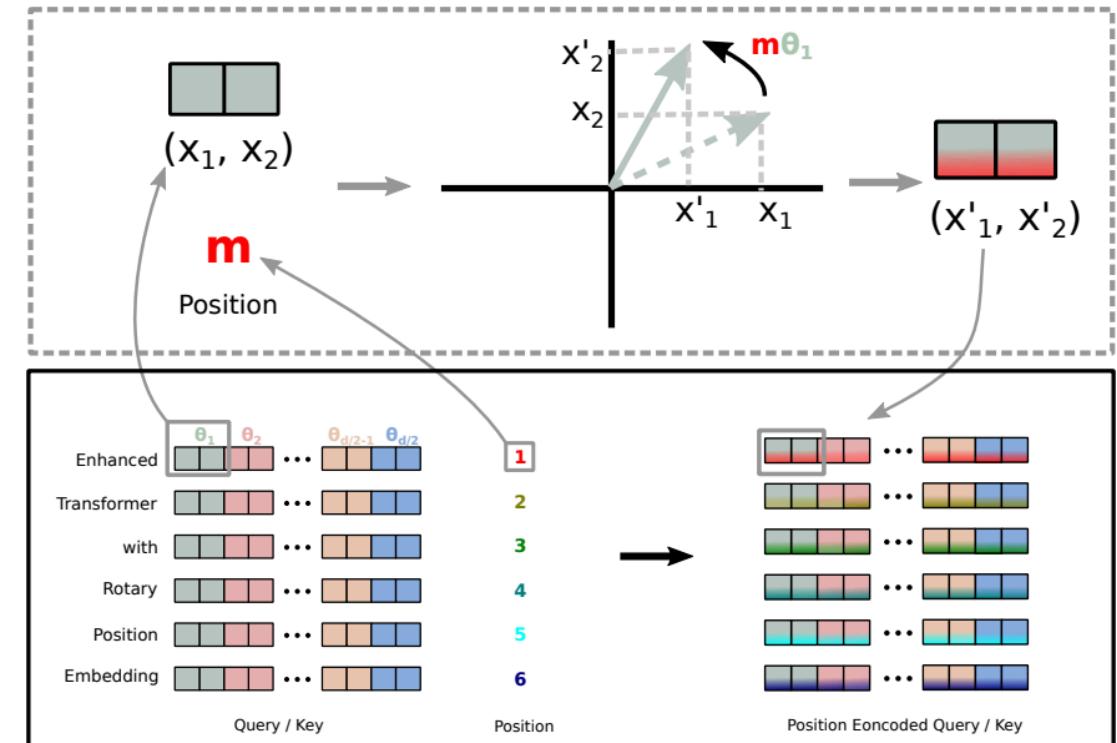


Figure 1: Implementation of Rotary Position Embedding(RoPE).

Decoder-only архитектура для генерации

От Bidirectional к Causal Attention

Проблема автогрессивной генерации

При генерации текста модель должна предсказывать следующий токен, **не видя будущих токенов**

Bidirectional (BERT)

"Кот сидит на ____"
↓ ↓ ↓ ↓
Видит все токены

Causal (GPT)

"Кот сидит на ____"
→ → → ?
Видит только предыдущие

Causal Masking: Реализация

Attention mask (нижнетреугольная матрица):

[1, 0, 0, 0] ← token 1 видит только себя
[1, 1, 0, 0] ← token 2 видит tokens 1-2
[1, 1, 1, 0] ← token 3 видит tokens 1-3
[1, 1, 1, 1] ← token 4 видит все предыдущие

Результат: Модель не может "подглядывать" в будущее → честная генерация

? Вопрос аудитории

Можно ли генерировать текст через BERT?



Ответ: Теоретически да, но это странно

🤔 Проблема с BERT

Архитектурные ограничения

- **Bidirectional attention** - видит весь контекст
- **Masked Language Modeling** - заполнение пропусков
- **Encoder-only** архитектура
- **Нет автогрессивной генерации**

Почему сложно?

BERT: [CLS] I love [MASK] [SEP]

↑ ↑ ↑ ↑

Видит все токены одновременно

💡 Возможные подходы

1. Iterative Refinement

1. Заменить все токены на [MASK]
2. Предсказать наиболее вероятные
3. Повторить для оставшихся [MASK]
4. Итерировать до конвергенции

2. BERT-based генераторы

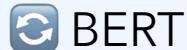
- **BART**: Encoder-decoder с BERT-like encoder
- **T5**: Text-to-text transformer
- **GLM**: General Language Model

3. Ограниченнная генерация

- **Заполнение шаблонов**
- **Sentence completion**
- **Paraphrasing**

GPT vs BERT: Архитектурные различия

Bidirectional vs Causal Attention



BERT

- Bidirectional encoder
- Masked language modeling
- Видит весь контекст одновременно
- Заполнение пропусков

"Кот MASK на столе"



GPT

- Unidirectional decoder
- Next token prediction
- Causal masking (только влево)
- Автогрессивная генерация

"Кот сидит → на → столе → ..."

Вывод: Архитектурный выбор определяет возможности модели

Эффективность генерации

Каждый шаг = полный forward pass

Step 1: "The" (prompt: "")

↓

Расчет attention для 1 токена

Step 2: "The weather" (prompt: "The")

↓

Расчет attention для 2 токенов

Step 3: "The weather is" (prompt: "The weather")

↓

Расчет attention для 3 токенов

Step 4: "The weather is nice" (prompt: "The weather is")

↓

Расчет attention для 4 токенов ...

Проблема: $O(n^2)$ на каждом шаге, без переиспользования вычислений



KV Cache: Технические детали

Key insight: При автогрессивной генерации предыдущие K,V не изменяются!

```
# Псевдокод KV Cache
def generate_with_cache(prompt):
    tokens = tokenize(prompt)
    kv_cache = [] # Сохраняем K,V для всех токенов

    for step in range(max_length):
        # Только для нового токена
        new_q, new_k, new_v = compute_qkv(tokens[-1])
        kv_cache.append((new_k, new_v))

        # Attention с полным кешем K,V
        output = attention(new_q, cached_keys, cached_values)
        next_token = sample(output)
        tokens.append(next_token)
```

Масштабирование контекста и эффективность

Проблема квадратичной сложности

Attention complexity: $O(n^2)$ по длине последовательности

Решения для длинных контекстов

- **Sparse attention patterns** - локальные окна внимания
- **Linear attention** - аппроксимация attention
- **Sliding window** - фиксированные окна
- **Dual Chunk Attention** (Qwen-1M)

Эволюция длины контекста

Модель

Длина контекста

GPT-1

512 токенов

GPT-2

1024 токенов

GPT-3

2048 токенов

GPT-4

8192 → 32768 токенов

Claude-3

200,000 токенов

Gemini-1.5

1,000,000 токенов

IV. Scaling Laws и современные подходы

Scaling Laws: Предсказуемые улучшения

Kaplan et al. (2020): Фундаментальное открытие

$$L(N, D, C) \propto N^{-\alpha} \cdot D^{-\beta} \cdot C^{-\gamma}$$

где:

- **L** - loss (качество модели)
- **N** - количество параметров
- **D** - размер датасета
- **C** - compute (вычислительные ресурсы)

Практические следствия

- **Предсказуемость:** Можно прогнозировать качество
- **Планирование:** Сколько качества будет на X долларов
- **Инвестиции:** ROI от увеличения масштаба

Chinchilla Scaling и compute-optimal training

Переосмысление соотношения параметров и данных

До Chinchilla

- **Больше параметров** → лучше качество
- Недостаточно данных для обучения
- Примеры: GPT-3 (300B токенов)

Chinchilla

- **Optimal ratio:** 20 токенов на параметр
- Меньше параметров, больше данных

После Chinchilla

LLaMA-3: 8B параметров, >15T токенов (1 875x)

QWEN3: 1.7B параметров, 36T токенов (27 692x)

Smaller models, more data > Larger models, less data

Emergent Abilities и Phase Transitions

Качественные скачки при масштабировании

Категория модели	Размер	Ключевые возможности
Малые модели	< 1B параметров	<ul style="list-style-type: none">• Базовые языковые навыки• Простая генерация• Ограниченнное понимание
Средние модели	1B - 10B параметров	<ul style="list-style-type: none">• Следование инструкциям• Базовое reasoning• Few-shot learning
Большие модели	> 10B параметров	<ul style="list-style-type: none">• Chain-of-thought reasoning• Code generation• Creative writing• Complex reasoning

Emergent Abilities и Phase Transitions (2025)

Качественные скачки при масштабировании

Категория модели	Размер	Ключевые возможности
Компакт-модели	< 8 В параметров	<ul style="list-style-type: none">• Базовые языковые и визуальные навыки• Энергоэффективный inference на устройстве• Узкофокусные суммаризации и поиск
Средние модели	3 В - 70 В параметров	<ul style="list-style-type: none">• Надёжное следование инструкциям• Multi-task reasoning & few-shot learning• Chain-of-thought + саморефлексия
Большие модели	70 В - 400 В параметров	<ul style="list-style-type: none">• Креативное многоязычное письмо• Глубокое мультимодальное понимание• reasoning
Фронтир-модели	> 400 В параметров	<ul style="list-style-type: none">• Планирование с инструментами и агенты• Многошаговое кросс-модальное решение задач

Направления развития в 2025+

- **Extended in-context learning** – 100 k+ токенов
- **Chain-of-thought с самокритикой** – self-reflection
- **Tool use / агентность** – API-вызовы, код, веб
- **Multimodal reasoning** – текст + изображения + аудио + видео
- **Code generation & repair** – автозапуск тестов, фиксы
- **Долговременная память** и персонализация

Современные архитектурные инновации

Mixture of Experts (MoE)

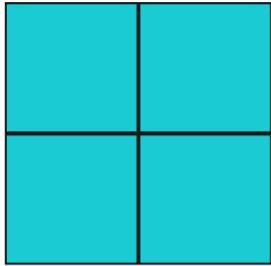
Идея: Активировать только часть параметров для каждого токена

$$\text{MoE Layer: } y = \sum_i G_i(x) \cdot Expert_i(x)$$

$G_i(x)$ - router network (softmax)

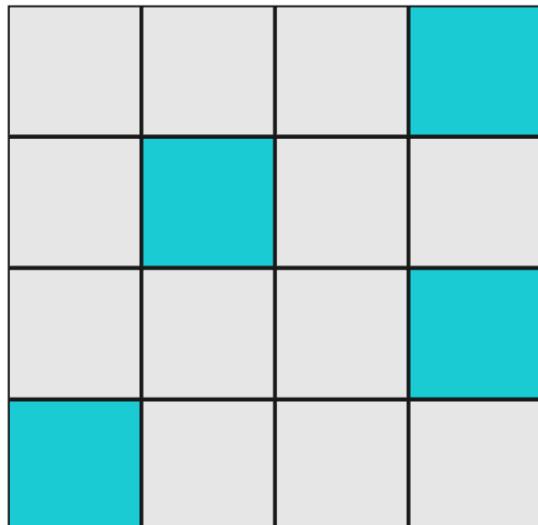
Преимущества: Больше параметров без пропорционального роста compute

Раньше мы всегда
использовали все
знания сразу

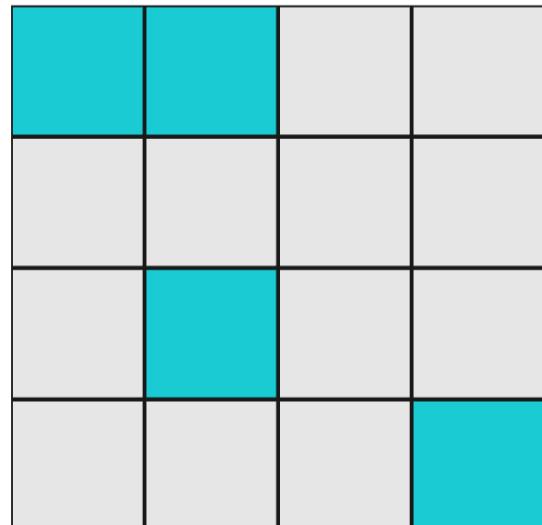


Теперь знаний гораздо больше, у
модели возможность выбирать нужную
ей информацию прямо сейчас

"Реши уравнение $x^2 = 4$ "



"Что приготовить на ужин?"



За счет этого, модели учатся
быстрее и лучше

Альтернативные архитектуры

Mamba

State Space Models как ответ на ограничения Transformer

Ключевые проблемы Transformer:

- **Квадратичная сложность:** $O(n^2)$ по длине последовательности
- **Memory bottleneck:** Растущие требования к памяти
- **Sequential generation:** Медленный inference при генерации

Решение Mamba:

- **Линейная сложность:** $O(n)$ вместо $O(n^2)$
- **Selective State Space:** Умное решение что запоминать/забывать
- **5x быстрее inference** по сравнению с Transformer

Текущее развитие (2024-2025):

- **Hybrid архитектуры:** Jamba = Mamba + Transformer

Diffusion Models: От изображений к тексту

Новый подход к генерации текста

Традиционные языковые модели генерируют текст автогрессивно - токен за токеном:

```
"Кот" → "сидит" → "на" → "столе"
```

Диффузионные модели работают принципиально иначе:

```
[NOISE] → Постепенное уточнение → "Кот сидит на столе"
```

Ключевые принципы

Процесс диффузии

1. **Начинаем с шума** вместо пустого контекста
2. **Итеративное уточнение** через несколько шагов
3. **Параллельная генерация** блоков текста
4. **Самокоррекция** на каждом шаге

Преимущества diffusion models

- ⚡ Более быстрая генерация (блоки vs токены)
- 🔄 Лучшая когерентность текста
- 🖌 Естественное редактирование в середине текста
- 🔧 Коррекция ошибок в процессе генерации

Gemini Diffusion (Google, 2025)

Экспериментальная модель от Google DeepMind, представленная на Google I/O 2025:

- 🚀 Скорость: 1479 токенов/сек (значительно быстрее предыдущих моделей)
- 🔄 Специализация: Особенно эффективна для редактирования кода и математики
- 🔄 Итеративная генерация: Улучшает качество через множественные проходы
- 📈 Производительность: Сопоставима с Gemini 2.0 Flash-Lite на большинстве бенчмарков

Диффузионные модели открывают новые возможности для более контролируемой и эффективной генерации текста

Landscape современных моделей

Закрытые модели

OpenAI GPT семейство

- **GPT-4o**: Optimized для speed/cost
- **o3**: Новейшая модель для reasoning
- **GPT-4.5**: Самая большая модель у OpenAI

Конкуренты

- **Claude-4** (Anthropic): Constitutional AI
- **Gemini-2.5** (Google): Native multimodal
- **Grok-4** (X): Ilon Musk's company

Open-source революция

- **LLaMA**: Эффективные модели от Meta
- **Mistral**: Европейские разработчики
- **Qwen**: Китайские модели, 140+ языков
- **Phi-4**: Microsoft, специализация на синте

V. Технические детали обучения

Tokenization: От текста к числам

Byte Pair Encoding (BPE)

Алгоритм BPE

Пример:

```
"hello" → ["he", "ll", "o", " "]
```

Шаги алгоритма:

1. Начинаем с отдельных символов
2. Итеративно объединяем наиболее частые пары
3. Повторяем до достижения нужного размера словаря

Результат: баланс между размером словаря и выразительностью

Преимущества BPE

- ✓ **Эффективность:** сокращение количества токенов
- ✓ **Обработка OOV:** новые слова разбиваются на знакомые части
- ✓ **Языковая универсальность:** работает с любыми языками
- ✓ **Морфологическая гибкость:** обработка словоформ

Используется в: везде

Pre-training: Самоконтролируемое обучение

Основная задача:

Предсказание следующего
токена

Loss функция:

$$\mathcal{L} = - \sum_{i=1}^n \log P(x_i | x_1, \dots, x_{i-1})$$

Пример:

Входная последовательность: "Кот сидел на"

Предсказание: "коврике" (вероятность 0.7)

Источники данных
(Petabytes)

Common Crawl - веб-страницы

Wikipedia - энциклопедические статьи

Books - Project Gutenberg, художественная
литература

Academic papers - научные публикации

Code repositories - GitHub, StackOverflow

Объем: триллионы токенов

Supervised Fine-tuning и Alignment

Supervised Fine-tuning (SFT)

Цель: обучение следованию инструкциям

Формат данных:

Input: "Объясни квантовую механику"

Target: "Квантовая механика описывает поведение
частиц на атомном уровне ... "

Источники данных:

- High-quality instruction-response pairs
- Человеческие демонстрации
- Синтетические данные

Reinforcement Learning from Human Feedback (RLHF)

Трехэтапный процесс:

1. **SFT:** обучение на качественных примерах
2. **Reward Modeling:** обучение оценки качества
ответов
3. **PPO Training:** оптимизация через reinforcement
learning

Memory Requirements: Реальные цифры

Пример: модель 70B параметров

Расчет памяти

Параметры модели (fp32):

$$70\text{B} \times 4 \text{ bytes} = 280 \text{ GB}$$

Градиенты (fp32):

$$70\text{B} \times 4 \text{ bytes} = 280 \text{ GB}$$

Optimizer states (fp32):

$$70\text{B} \times 8 \text{ bytes} = 560 \text{ GB}$$

Activations: ? (batch dependent)

Итого: >1 120 GB

Практические выводы

Обычное железо:

- RTX 4090: 24 GB (недостаточно)
- Игровой ПК: 64 GB RAM (недостаточно)

Требуется:

- $16 \times \text{A100 80GB} = 1280 \text{ GB}$
- Или $16 \times \text{H100 80GB} = 1280 \text{ GB}$

Стоимость обучения Qwen 2.5 70B:

миллионы долларов

Distributed Training: Современные подходы

Стратегии параллелизма

Data Parallelism

GPU 1: Batch 1 → Forward → Backward

GPU 2: Batch 2 → Forward → Backward

GPU 3: Batch 3 → Forward → Backward

↓

All-reduce градиентов

Model Parallelism

GPU 1: Layers 1-10

GPU 2: Layers 11-20

GPU 3: Layers 21-30

Optimization Techniques

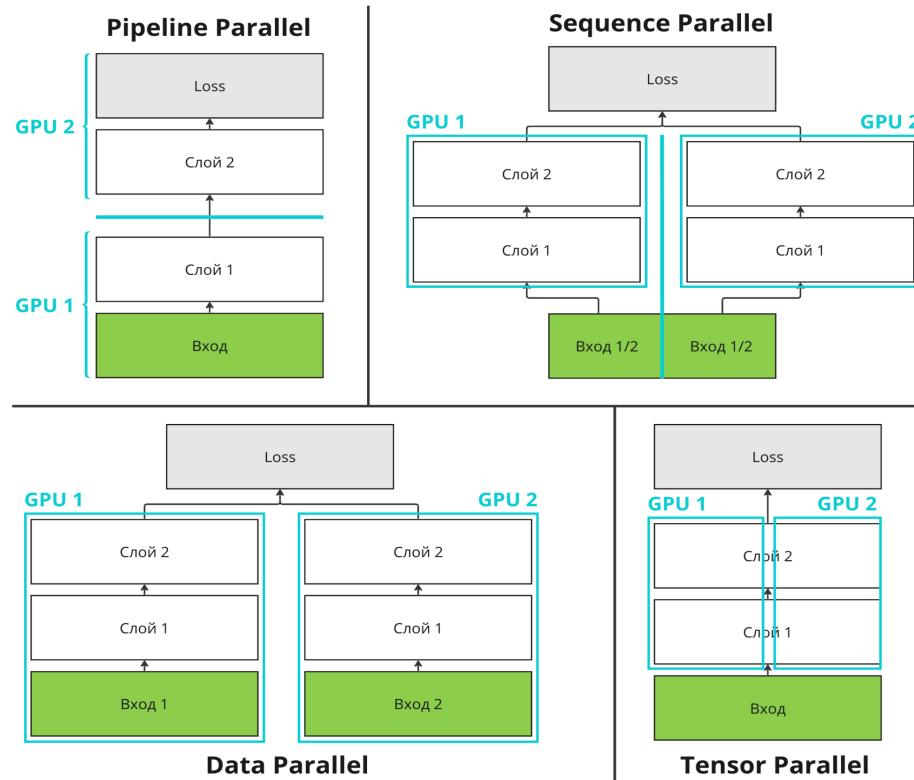
Memory Optimization

- **Mixed Precision:** fp16 + fp32 для стабильности
- **Gradient Checkpointing:** пересчет вместо хранения

Communication Optimization

- **Gradient Compression:** сжатие градиентов
- **Overlapped Communication:** параллельные операции
- **Hierarchical AllReduce:** оптимизация сети

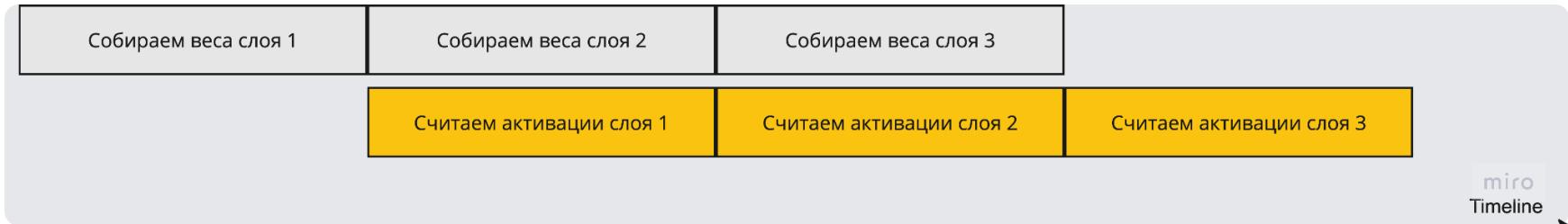
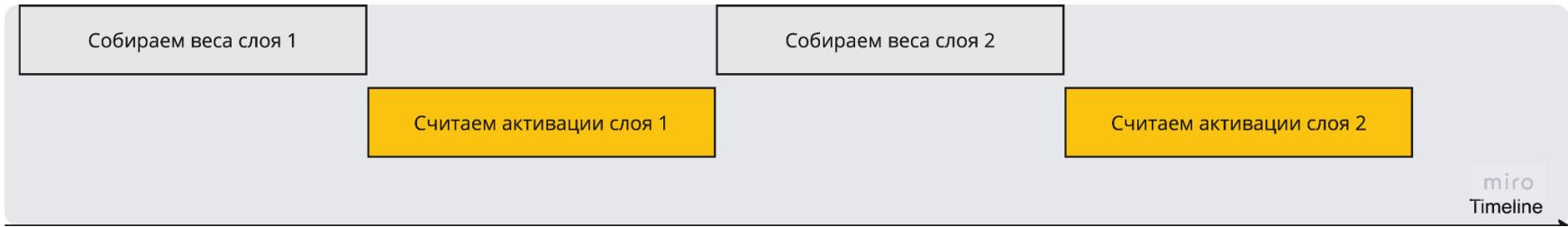
Distributed Training: наглядно



Distributed Training: ZeRO



Distributed Training: ZeRO



Современные фреймворки

DeepSpeed (Microsoft)

- ZeRO optimizer (stages 1-3)
- Trillion-parameter training
- 3D parallelism support

Megatron-LM (NVIDIA)

- Tensor + Pipeline + Data parallelism
- Optimized for large models
- Advanced memory management

PyTorch FSDP (Meta)

- Fully Sharded Data Parallel
- Native PyTorch integration
- Hybrid sharding strategies

Unslot (Community)

- 2x faster fine-tuning, 70% less memory
- Consumer GPU support (GTX 1070+)
- Optimized LoRA/QLoRA implementation

Выбор стратегии обучения

В зависимости от размера модели

Для моделей <10B параметров:

- **Standard DDP** (Distributed Data Parallel)
- **Single-node training** на мощных серверах

Для моделей 10B-100B параметров:

- **FSDP** (Fully Sharded Data Parallel) или **DeepSpeed ZeRO-2/3**
- **Multi-node training** на кластере

Для моделей >100B параметров:

- **3D parallelism** (data + model + pipeline)
- **Pipeline parallelism** для больших кластеров
- **Aggressive optimization** всех компонентов

Оптимизация производительности

Ключевые аспекты для эффективного обучения

Memory Efficiency

- **Gradient checkpointing** - экономия памяти за счет пересчета
- **Mixed precision (bf16)** - снижение требований к памяти
- **Optimal batch size** - максимальная утилизация GPU
- **ZeRO optimizer** - партиционирование состояний

Communication Efficiency

- **Computation/communication overlap** - параллельная работа
- **Efficient backends** - NCCL для NVIDIA, Gloo для CPU
- **Network topology optimization** - учет архитектуры кластера
- **Gradient compression** - сокращение трафика

Мониторинг и профилирование

- **GPU utilization** - эффективность использования вычислителей
- **Memory usage** - контроль потребления памяти
- **Communication overhead** - анализ сетевых задержек

Заключение

Ключевые инсайты

Эволюция обучения: Pre-training → SFT → RL → Production-ready model

Главные вызовы:

- Computational requirements
- Memory limitations
- Communication bottlenecks

Решения:

- Parameter-efficient methods (LoRA, QLoRA)
- Distributed training frameworks
- Advanced optimization techniques

Будущее: демократизация обучения больших моделей на специальных чипах для AI

VI. Передовые методы и техники

Model Efficiency и Optimization

Quantization

Post-training Quantization (PTQ)

FP32 (32-bit) → INT8 (8-bit) → INT4 (4-bit)

Размер модели: 4x–8x меньше

Speed: 2–4x быстрее

Качество: -1–3% точности

Популярные методы квантизации

- **GPTQ:** Оптимальная квантизация для LLM
- **AWQ (Activation-aware Weight Quantization):** Учёт важности весов
- **GGUF/GGML:** Эффективный формат для CPU inference

Parameter-Efficient Fine-tuning

LoRA (Low-Rank Adaptation)

Основная идея

Формула:

$$W' = W_0 + \Delta W = W_0 + BA$$

где:

- $B \in \mathbb{R}^{(n \times r)}$, $A \in \mathbb{R}^{(r \times m)}$
- $r \ll \min(n, m)$ - ранг адаптации
- W_0 - замороженные веса

Экономия памяти: ~90%

Продвинутые методы

QLoRA: 4-bit quantization + LoRA

- 4-bit NormalFloat (NF4) для весов
- Double quantization констант
- Paged optimizers

AdaLoRA: Adaptive rank allocation

- Динамическое распределение ранга
- Важные слои получают больший ранг

Результат: 65B модель на 48GB GPU

Flash Attention и оптимизация внимания

Flash Attention v2/v3

Проблема: Квадратичная сложность внимания $O(n^2)$

Решение: IO-aware алгоритм

- Tiling для эффективного использования SRAM
- Fused kernels без материализации матрицы внимания
- Speedup: 2-4x, меньше памяти

Альтернативные механизмы внимания

- **Multi-Query Attention (MQA):** Общие K, V для всех голов
- **Grouped-Query Attention (GQA):** Компромисс между MHA и MQA
- **Sliding Window Attention:** Локальный контекст (Mistral)
- **Ring Attention:** Распределённое внимание для длинных последовательностей

Knowledge Distillation и Model Compression

Teacher-Student Framework

Цель: Передать знания большой модели маленькой

$$\mathcal{L}_{total} = \alpha \mathcal{L}_{CE} + (1 - \alpha) \mathcal{L}_{KL}(p_{teacher}, p_{student})$$

где:

- \mathcal{L}_{CE} - cross-entropy loss на истинных метках
- \mathcal{L}_{KL} - KL divergence между распределениями
- Temperature scaling для soft targets

Современные подходы к дистилляции

- **Progressive Distillation:** Постепенное уменьшение модели
- **Self-Distillation:** Модель учится у себя же
- **Cross-Architecture Distillation:** Между разными архитектурами
- **Task-Specific Distillation:** Оптимизация под конкретную задачу

Успешные примеры (2024-2025)

- **Llama 3.2 (1B, 3B):** Дистиллированы из Llama 3.1 8B
- **Gemma 2 (2B, 9B):** Эффективные модели от Google
- **Qwen2.5 (0.5B-7B):** Семейство моделей разных размеров

Multimodal Integration

От текста к мультимодальным моделям

Ограничения текстовых моделей

- Только текстовая информация
- Нет понимания визуального контекста
- Ограниченнное представление о мире
- Сложность описания изображений/видео

Мультимодальные возможности

- **Vision + Language:** GPT-4o, Gemini Pro Vision
- **Speech + Text:** Whisper, voice assistants
- **Video understanding:** анализ видеоконтента
- **Real-world:** связь с физическим миром

Архитектурные подходы

- **Early fusion:** Объединение на уровне входных данных
- **Late fusion:** Независимая обработка, затем объединение
- **Cross-modal attention:** Взаимодействие между модальностями
- **Unified representations:** Общее пространство для всех модальностей

Современные мультимодальные модели

Vision-Language Models

GPT-4 Vision / GPT-4o

- Анализ изображений и текста
- Описание сцен, чтение диаграмм
- OCR и понимание документов
- Интерактивные диалоги об изображениях

Gemini Pro Vision

- Native multimodal architecture
- Обработка длинных видео
- Пространственное понимание
- Интеграция с Google Services

LLaVA (Large Language-and-Vision Assistant)

- Open-source альтернатива
- Instruction-following для изображений
- Эффективное обучение на образах + тексте

DALL-E / Midjourney / Stable Diffusion

- Text-to-image генерация
- Креативные задачи
- Редактирование изображений
- Стилизация и адаптация

Мультимодальные модели: вызовы

- **Alignment модальностей:** синхронизация представлений
- **Scale mismatch:** разные объемы данных для модальностей
- **Computational efficiency:** обработка больших изображений/видео
- **Evaluation:** сложность оценки мультимодального качества

Спасибо за внимание!

Вопросы и обсуждение

Генеративные модели продолжают революционизировать NLP. Ключ к успеху - понимание фундаментальных принципов и активное участие в развитии области.

Удачи в изучении и применении генеративных моделей! 