

# **Профилирование и сравнительный анализ производительности моделей автоматического распознавания речи (ASR)**

## **Команда:**

Евгений Романов (tg: @wallrich)  
Эмиль Деникаев (tg: @emildenikaev)  
Фанис Нигамадянов (tg: @FanisNgyv)  
Ева Левченко (tg: @eva\_lions)

17 ноября 2025 г.

# **Содержание**

<b>1 Введение</b>	<b>4</b>
1.1 Мотивация . . . . .	4
1.2 Цели и задачи исследования . . . . .	4
1.3 Объекты исследования . . . . .	4
<b>2 Методология эксперимента</b>	<b>4</b>
<b>3 Результаты и анализ</b>	<b>4</b>
3.1 Сравнительный анализ производительности . . . . .	4
3.2 Анализ "узких мест" по результатам профилирования . . . . .	6
<b>4 Предлагаемые пути оптимизации</b>	<b>6</b>
<b>5 Заключение</b>	<b>6</b>
5.1 Итоговые выводы . . . . .	6
5.2 Дальнейшие шаги . . . . .	7

## Аннотация

В данной работе проведён комплексный анализ производительности и эффективности пяти популярных моделей автоматического распознавания речи (ASR): четырёх вариаций OpenAI Whisper и Wav2Vec2 от Facebook. Исследование направлено на изучение масштабируемости моделей в зависимости от длины аудио, размера пакета и используемого аппаратного обеспечения (CPU vs. GPU). Ключевые метрики, включая задержку, энергопотребление и качество распознавания (Word Error Rate), были агрегированы и проанализированы для прямого сравнения.

Результаты демонстрируют значительное превосходство GPU над CPU в скорости и энергоэффективности. Модель Wav2Vec2 выделяется как наиболее производительное решение для производственных систем. С увеличением размера моделей Whisper наблюдается значительный рост требований к ресурсам. На основе детального профилирования были выявлены основные вычислительные "узкие места" и предложены конкретные пути их оптимизации.

# 1 Введение

## 1.1 Мотивация

Автоматическое распознавание речи (ASR) стало неотъемлемой частью современных технологий. С ростом сложности моделей вопросы производительности, задержки и энергопотребления выходят на первый план. Для приложений реального времени, облачных сервисов и встраиваемых систем критически важно понимать, как ведут себя различные модели в разных условиях, чтобы сделать осознанный выбор и определить стратегию оптимизации.

## 1.2 Цели и задачи исследования

Основная цель данной работы — изучить, как масштабируются популярные ASR-модели, и выявить их слабые места для дальнейшей оптимизации. Задачи включали сравнительный анализ производительности на CPU и GPU, оценку влияния длины аудио и размера батча, измерение ключевых метрик (задержка, энергопотребление, WER) и определение наиболее ресурсоемких операций.

## 1.3 Объекты исследования

- **Модели:** openai-whisper-tiny, -base, -small, -medium и facebook/wav2vec2-base-960h.
- **Датасет:** speechcolab/gigaspeech (валидационный набор).
- **Аппаратное обеспечение:** Google Colaboratory (CPU и GPU Tesla P100).

# 2 Методология эксперимента

Для обеспечения воспроизводимости тестов был реализован механизм подготовки и кеширования аудиофрагментов точной целевой длины (5, 15, 30 и 60 секунд). Задержка измерялась с помощью библиотеки `time` с вызовом `torch.cuda.synchronize()` для GPU. Энергопотребление отслеживалось библиотекой `codecarbon`, а качество (WER) — `jiwer`. Для детального анализа операций использовался встроенный профайлер `torch.profiler`.

# 3 Результаты и анализ

## 3.1 Сравнительный анализ производительности

Ключевые результаты экспериментов представлены в Таблице 1 для GPU и Таблице 2 для CPU. Данные сгруппированы по метрикам, что позволяет удобно сравнивать модели между собой.

Таблица 1: Сравнительные метрики производительности на GPU.

Метрика (GPU)	tiny	base	small	medium	wav2vec2
<i>Задержка (мс, batch=1, 60с аудио)</i>					
	277.2	402.1	724.4	1307.6	<b>204.0</b>
<i>Энергия (Дж/сэмпл, batch=16, 60с аудио)</i>					
	0.28	0.55	1.59	4.77	<b>0.11</b>
<i>Макс. проп. способность (сэмплов/сек)</i>					
	54.1	26.2	28.4	12.9	<b>95.4</b>
<i>Качество (WER, %)</i>					
	36.0	34.0	36.0	36.0	57.0

Выводы по GPU:

- **Wav2Vec2 — лидер производительности:** Модель от Facebook демонстрирует самую низкую задержку, наименьшее энергопотребление и самую высокую пропускную способность, что делает ее абсолютным чемпионом для производственных систем.
- **Цена качества Whisper:** С увеличением размера моделей Whisper от `tiny` до `medium` задержка возрастает почти в 5 раз, а энергопотребление — в 17 раз. Это наглядно показывает, насколько дорогим с вычислительной точки зрения является повышение качества транскрипции.
- **Компромисс Whisper-tiny/base:** Младшие модели Whisper представляют собой хороший баланс, значительно опережая по скорости старшие модели, но уступая Wav2Vec2.

Таблица 2: Сравнительные метрики производительности на CPU.

Метрика (CPU)	tiny	base	small	medium	wav2vec2
<i>Задержка (мс, batch=1, 60с аудио)</i>					
	1046.3	2086.5	5889.4	16231.5	7384.7
<i>Энергия (Дж/сэмпл, batch=16, 60с аудио)</i>					
	4.8	11.3	36.0	129.2	74.6

Выводы по CPU:

- **Непригодность для production:** Производительность на CPU на порядок ниже, чем на GPU. Задержки, измеряемые секундами и десятками секунд, делают использование ASR-моделей на CPU непрактичным для большинства реальных задач.
- **Wav2Vec2 теряет преимущество:** Интересно, что на CPU модель Wav2Vec2, хоть и быстрее крупных моделей Whisper, но медленнее `whisper-tiny` и `-base`. Это говорит о том, что ее архитектура сильно оптимизирована именно под параллельные вычисления на GPU.

### 3.2 Анализ "узких мест" по результатам профилирования

Детальное профилирование с помощью `torch.profiler` позволило выявить наиболее затратные операции, которые являются главными кандидатами на оптимизацию.

**Для моделей Whisper:** Главными вычислительными "узкими местами" являются операции **матричного умножения** (`aten::addmm`), составляющие от 15% до 52% времени выполнения на GPU, и **механизмы внимания** (`...attention`), занимающие от 25% до 47% времени. Это полностью коррелирует с их вычислительной сложностью: энкодер `whisper-medium` требует **917.5 GFLOPS**, в то время как у `whisper-tiny` этот показатель составляет всего **23.2 GFLOPS**. Также стоит отметить значительные накладные расходы на операции выделения памяти (`aten::empty`).

**Для модели Wav2Vec2:** Архитектура Wav2Vec2 в большей степени опирается на **сверточные операции** (`cudnn_convolution`), которые занимают около 44% времени GPU. На втором месте стоит **матричное умножение** ( $\sim 34\%$ ).

**Выводы из профилирования:** Оптимизация производительности ASR-моделей должна быть сфокусирована на ускорении базовых операций глубокого обучения: матричного умножения и сверток. Снижение накладных расходов на выделение памяти и запуск CUDA-ядер также является важным направлением для улучшения.

## 4 Предлагаемые пути оптимизации

На основе анализа "узких мест" можно предложить следующие стратегии:

- **Оптимизации для GPU:** Увеличение размера батча, использование Tensor Cores (смешанная точность FP16), кэширование вычислений.
- **Оптимизации кода:** Применение *Kernel Fusion* для слияния мелких операций (например, LayerNorm + Linear), квантование (FP16/INT8) для ускорения инференса.
- **Системные оптимизации:** Использование CUDA Graphs для устранения накладных расходов на запуск ядер, применение Stream Parallelism для перекрытия вычислений и передачи данных.

**Выводы по оптимизации:** Наибольший и самый быстрый выигрыш в производительности для всех моделей на GPU дает увеличение размера батча. Для дальнейшего ускорения следует применять более сложные техники, такие как квантование и слияние ядер.

## 5 Заключение

### 5.1 Итоговые выводы

Исследование показало, что выбор ASR-модели является компромиссом между качеством, скоростью и потребляемыми ресурсами. Wav2Vec2 является бесспорным лидером по производительности, в то время как семейство Whisper предлагает градации качества с соответствующим ростом вычислительных затрат. Для эффективной работы любой из рас-

смотренных моделей в производственной среде необходимо использование GPU и техник пакетной обработки.

Итоговые рекомендации по выбору модели сведены в Таблицу 3.

Таблица 3: Итоговые рекомендации по выбору модели.

Сценарий использования	Рекомендуемая модель	Обоснование
Production-системы, real-time	Wav2Vec2	Лучшее сочетание скорости и энергоэффективности.
Сбалансированные приложения	Whisper-tiny / -base	Хороший компромисс между скоростью и качеством.
Высококачественная транскрипция	Whisper-small / -medium	Максимальное качество, но требует значительных ресурсов и оптимизации.
Edge-устройства	Whisper-tiny + Quantization	Низкие требования к памяти и ресурсам.

## 5.2 Дальнейшие шаги

Для дальнейшего развития проекта можно наметить следующие шаги:

- **Практическое применение оптимизаций:** Реализовать INT8-квантование для моделей Whisper с помощью `torch.quantization` и измерить прирост производительности и возможное падение качества (WER).
- **Анализ декодера:** Провести более глубокое профилирование именно авторегрессионного декодера моделей Whisper, так как он является основным источником задержек при генерации длинных текстов. Исследовать эффективность KV-кэширования.
- **Расширение набора моделей:** Включить в сравнение современные архитектуры, такие как Conformer или модели из фреймворка NeMo от NVIDIA, чтобы получить более полную картину состояния современных ASR-систем.