Профилирование ASR-моделей на TEDLIUM: латентность, энергопотребление и узкие места

Корнилова Валерия Александровна

29 ноября 2025 г.

Аннотация

В отчёте представлены результаты профилирования моделей автоматического распознавания речи (ASR) на датасете TEDLIUM Release 3 (dev set). Эксперименты проведены для моделей семейства Whisper (`whisper-tiny`, `whisper-base`, `whisper-small`, `whisper-medium`, `whisper-large-v3-turbo`) и `facebook/wav2vec2-base-960h` на двух типах устройств: CPU и GPU T4 (Google Colab). Измерялись латентность (latency), real-time factor (RTF), качество (WER/CER), энергопотребление и оценочные FLOPs, а также разбиение времени по стадиям пайплайна (препроцессинг, энкодер, декодер/CTC). На основе экспериментальных данных сделаны выводы о компромиссе скорость/качество/энергия и определены узкие места, требующие оптимизации с точки зрения времени и энергопотребления.

Содержание

1 Введение

Цель работы — исследовать, как масштабируются современные модели автоматического распознавания речи по длине аудио, размеру батча и типу устройства, и какие части моделей являются основными источниками задержки и энергопотребления.

Основные вопросы:

- Как растёт латентность с увеличением длины аудио и batch size для разных моделей?
- Как соотносятся качество (WER/CER) и задержка (latency / RTF)?
- Насколько сильно GPU выигрывает у CPU по времени и как это отражается на энергозатратах?
- Какие стадии пайплайна (препроцессинг, энкодер, декодер/CTC) дают наибольший вклад во время и энергию?

Ответы на эти вопросы важны для выбора модели под конкретный сценарий (онлайн/оффлайн, ограниченные ресурсы, требования к качеству) и для планирования дальнейших оптимизаций.

2 Модели и датасет

2.1 Набор моделей

В экспериментах участвовали модели:

- **Whisper (OpenAI)**:
  - `openai/whisper-tiny`,
  - `openai/whisper-base`,
  - `openai/whisper-small`,
  - `openai/whisper-medium`,
  - `openai/whisper-large-v3-turbo`.
- **Wav2Vec2 (Meta)**:
  - `facebook/wav2vec2-base-960h`.

Whisper — энкодер–декодер с авторегрессионным декодированием через `generate()`. `wav2vec2-base-960h` — CTC-модель: энкодер + линейная голова и CTC-декодирование (greedy).

2.2 Датасет TEDLIUM Release 3 (dev)

Используется dev-набор TEDLIUM Release 3:

- аудио: 16 кГц, моно;
- транскрипции (английский язык);
- флаг `ignore_time_segment_in_scoring` для исключаемых сегментов.

Аудио загружается через `datasets` (HuggingFace) и декодируется в `numpy`-массивы через `soundfile`/`librosa`. Для каждого примера вычисляется длительность.

2.3 Бакеты по длине и batch size

Бакеты по длине аудио:

- **5 s**: приблизительно 3–7 секунд;
- **15 s**: 10–20 секунд;
- **30 s**: 25–40 секунд;
- (опционально) **60 s**: 50–75 секунд.

Размер батча: batch_size $\in \{1, 4, 8, 16, 32\}$.
Для каждой комбинации (модель, устройство, длина, batch size) выбирается фиксированное число примеров, чтобы результаты были сопоставимы.

3 Методика эксперимента

3.1 Устройства

Эксперименты выполнены в двух конфигурациях:

- **CPU**: Google Colab без GPU, устройство `cpu`.
- **GPU**: Google Colab с GPU T4, устройство `cuda`.

Результаты каждого прогона сохраняются в CSV:

- `asr_profiling_cpu_only.csv`,
- `asr_profiling_gpu_only.csv`.

3.2 Метрики

По каждой конфигурации (`model_name`, `device_type`, `length_bucket`, `batch_size`) измеряются:

- **Latency per utterance, ms** (`latency_ms_per_utt`);
- **RTF (Real-Time Factor)** (`rtf`);
- **WER** и **CER** (через `jiwer`);
- **Энергия**:
  - энергия на конфигурацию, на utterance, на секунду аудио (`energy_j_per_utt`, `energy_j_per_audio_sec`);
  - расчёт — мощность CPU/GPU $\times$ wall-clock время;
- **FLOPs** (через `thop`/`fvcore`, `gflops_per_example`);
- **Максимальная видеопамять** (для GPU; `max_gpu_mem_bytes`);
- **Разбиение времени по стадиям**:
  - `t_preproc_ms_per_utt_config` — препроцессинг (извлечение признаков),
  - `t_encoder_ms_per_utt_config` — энкодер,
  - `t_decoder_ms_per_utt_config` — декодер / CTC.

Далее метрики усредняются по всем конфигурациям (всем длинам и batch size) для каждой модели и устройства.

4 Качество распознавания (WER/CER)

4.1 Средний WER/CER по моделям

Средний WER и CER (усреднение по всем конфигурациям и устройствам) приведены в табл. 1.

Модель	WER, %	CER, %
`openai/whisper-large-v3-turbo`	11.9	5.6
`openai/whisper-small`	14.8	8.2
`openai/whisper-medium`	14.8	8.2
`openai/whisper-base`	18.0	10.4
`openai/whisper-tiny`	22.0	12.2
`facebook/wav2vec2-base-960h`	22.6	7.3

Таблица 1: Средний WER/CER по моделям на TEDLIUM dev (по всем длинам и batch size).

Выводы:

- наилучшее качество показывает `whisper-large-v3-turbo` (WER $\approx 11.9\%$);
- `whisper-small` и `whisper-medium` дают сходный WER $\approx 14.8\%$;
- `whisper-base` и особенно `whisper-tiny` заметно проигрывают по WER;
- `wav2vec2-base-960h` имеет WER сопоставимый с `whisper-tiny`, при относительно невысоком CER.

4.2 WER vs длина аудио

Зависимость WER от длины аудио для GPU и CPU показана на рис. 1 и 2.

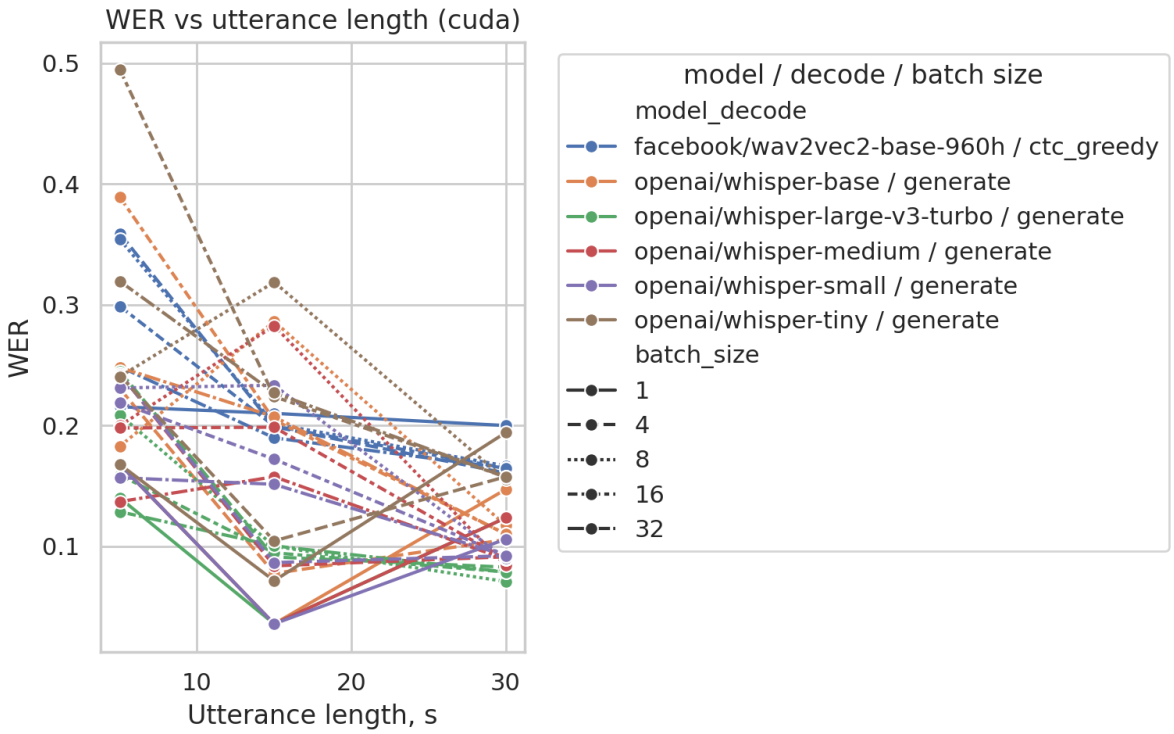


Рис. 1: WER vs длина аудио (device = cuda).

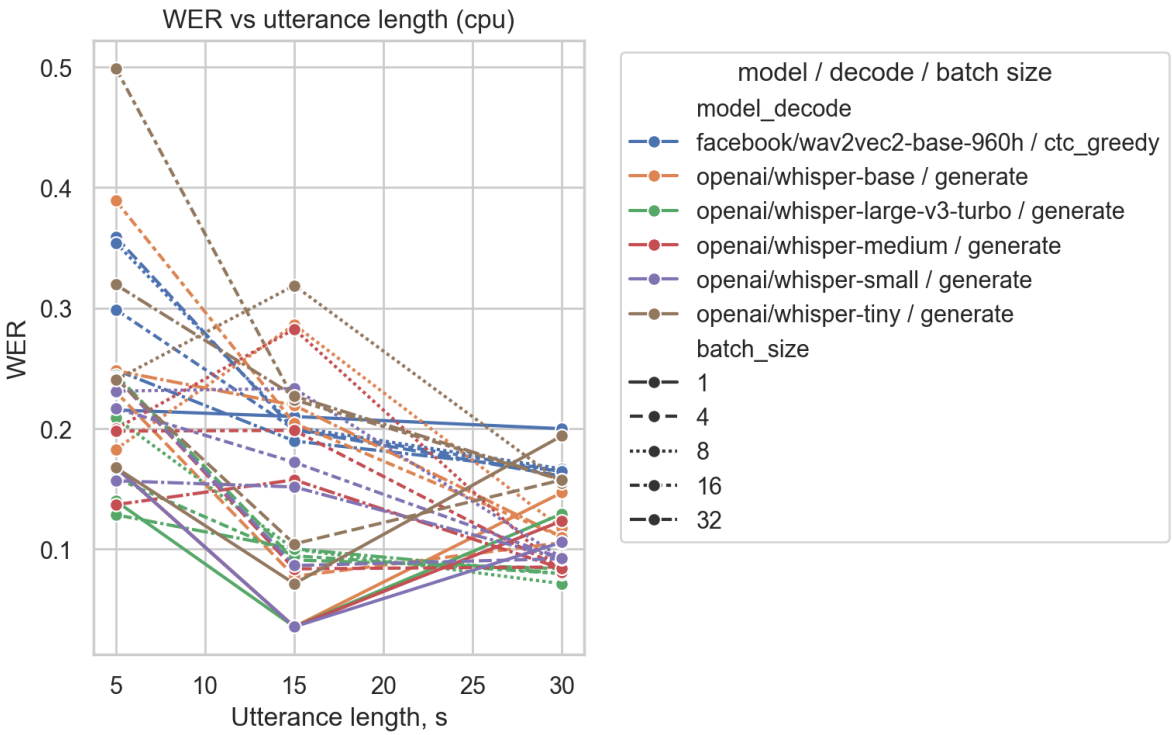


Рис. 2: WER vs длина аудио (device = cpu).

Наблюдения:

- WER слегка растёт с длиной аудио для всех моделей;
- относительный порядок моделей по качеству сохраняется;
- различий между CPU и GPU по WER нет (это одно и то же состояние моделей).

5 Латентность и RTF

5.1 Средние значения по моделям и устройствам

На основе CSV были получены средние значения latency, RTF и энергии по каждой модели и устройству (усреднение по всем длинам и batch size). Ниже приводятся ключевые цифры.

Для CPU (усреднённо):

- `wav2vec2-base-960h`: latency $\approx$ 419 ms/utt, RTF $\approx$ 0.03;
- `whisper-tiny`: $\approx$ 376 ms/utt, RTF $\approx$ 0.033;
- `whisper-base`: $\approx$ 658 ms/utt, RTF $\approx$ 0.058;
- `whisper-small`: $\approx$ 1663 ms/utt, RTF $\approx$ 0.131;

- `whisper-large-v3-turbo`: ≈ 2930 ms/utt, RTF ≈ 0.28;
- `whisper-medium`: ≈ 4344 ms/utt, RTF ≈ 0.353.

Для GPU (T4):

- `wav2vec2-base-960h`: latency ≈ 103 ms/utt, RTF ≈ 0.0069;
- `whisper-tiny`: ≈ 209 ms/utt, RTF ≈ 0.0186;
- `whisper-base`: ≈ 261 ms/utt, RTF ≈ 0.0206;
- `whisper-small`: ≈ 393 ms/utt, RTF ≈ 0.0287;
- `whisper-large-v3-turbo`: ≈ 322 ms/utt, RTF ≈ 0.0283;
- `whisper-medium`: ≈ 772 ms/utt, RTF ≈ 0.0555.

Относительное ускорение GPU относительно CPU по latency и RTF:

Модель	Ускорение latency	Ускорение RTF	Энергия(GPU)/Энергия(CPU)
`wav2vec2-base-960h`	4.1×	4.3×	≈ 3.3×
`whisper-base`	2.5×	2.8×	≈ 4.4×
`whisper-large-v3`	9.1×	9.9×	≈ 1.2×
`whisper-medium`	5.6×	6.4×	≈ 1.9×
`whisper-small`	4.2×	4.6×	≈ 2.6×
`whisper-tiny`	1.8×	1.8×	≈ 6.3×

То есть:

- GPU даёт ускорение от 1.8× (tiny) до 9× (large-v3) по latency;
- особенно сильно выигрывают крупные модели, для которых CPU оказывается слишком медленным.

5.2 Latency vs длина аудио

Зависимость latency от длины аудио (лог-шкала по оси Y) показана на рис. 3 и 4.

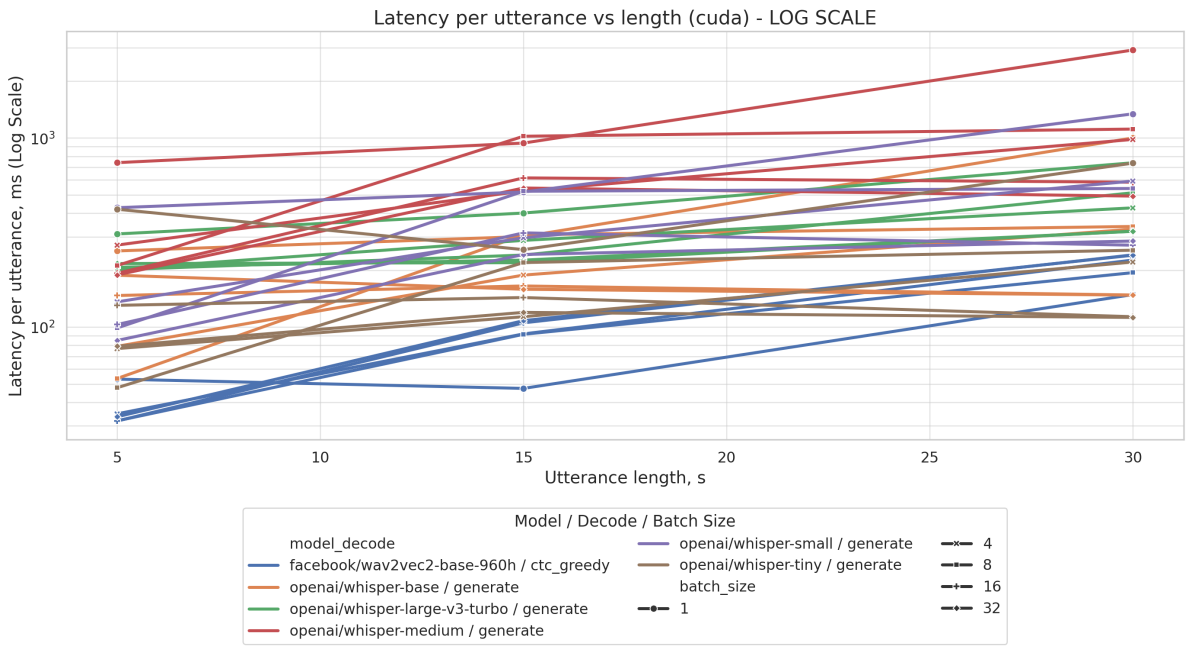


Рис. 3: Latency per utterance vs длина аудио (device = cuda, логарифмическая шкала).

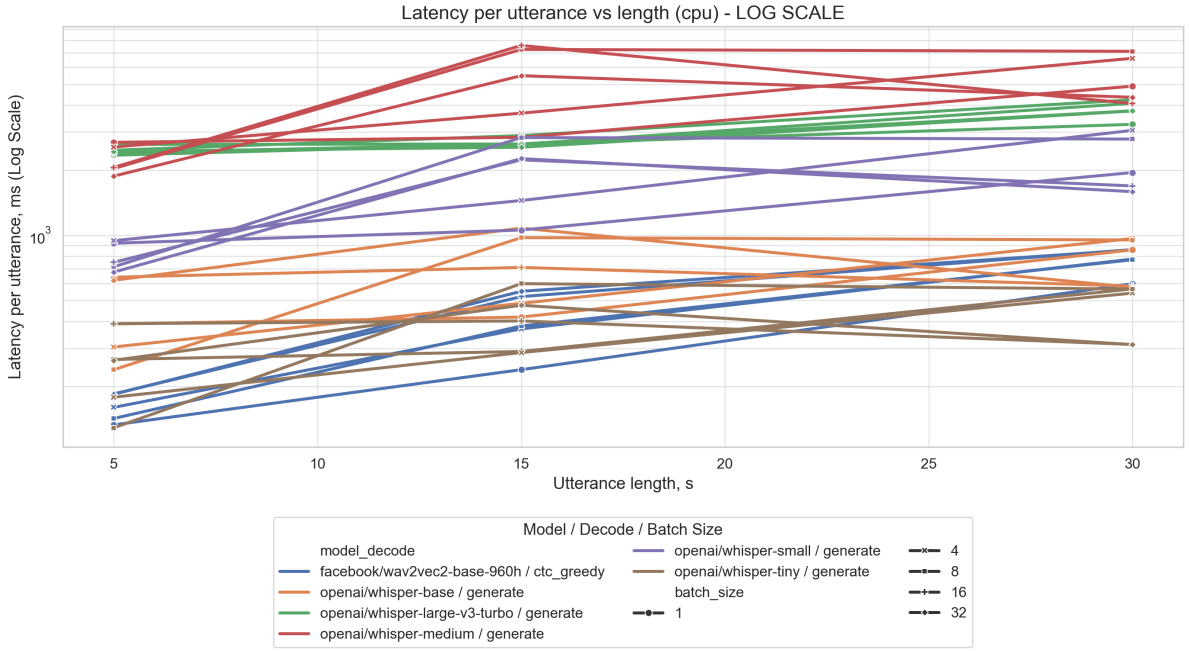


Рис. 4: Latency per utterance vs длина аудио (device = cpu, логарифмическая шкала).

Наблюдения:

- latency растёт почти линейно с длиной аудио для всех моделей;
- более крупные модели (medium, large-v3) имеют более крутой наклон;

- на GPU даже крупные модели могут работать с приемлемой latency для 15–30-секундных фрагментов, тогда как на CPU время обработки вырастает до нескольких секунд на utterance.

5.3 WER vs RTF

Компромисс “качество / скорость относительно real-time” для GPU иллюстрирует рис. 5.

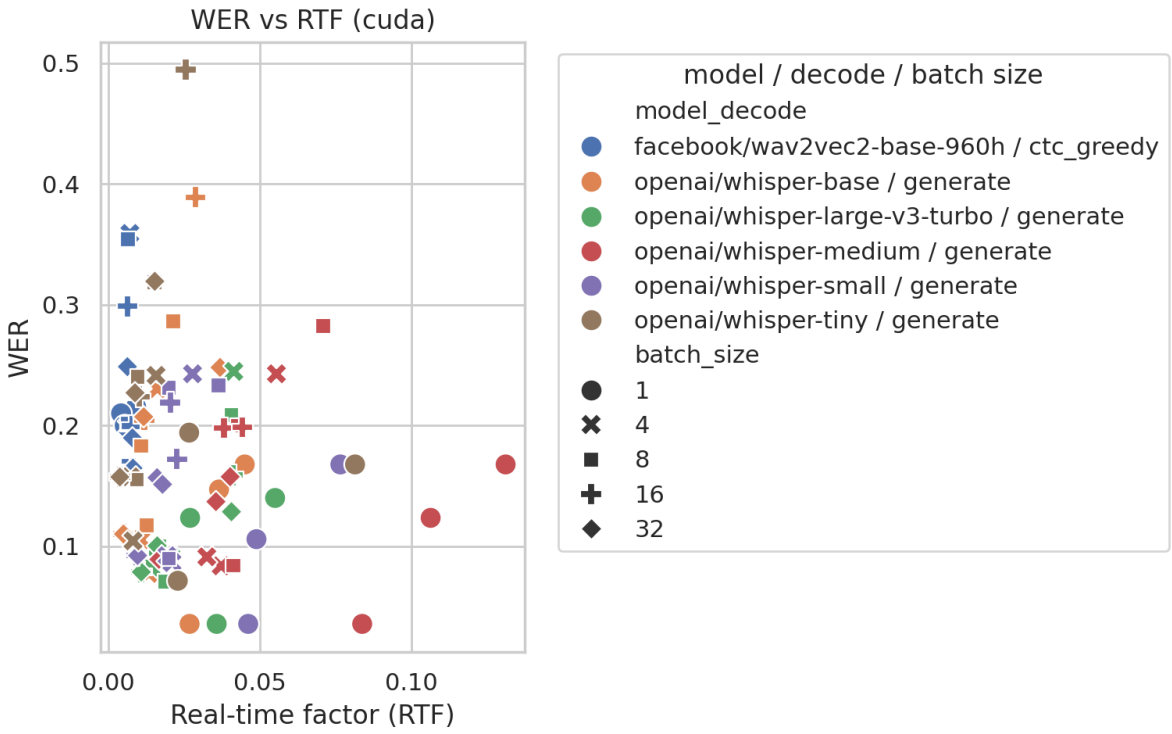


Рис. 5: WER vs RTF (device = cuda).

С точки зрения RTF:

- на GPU большинство моделей демонстрируют RTF < 0.1, то есть работают быстрее реального времени даже на достаточно длинных фрагментах;
- на CPU RTF у крупных моделей (`whisper-medium`, `whisper-large-v3`) $\gg 0.1$, обработка заметно медленнее реального времени.

6 Энергопотребление

6.1 Средняя энергия по моделям и устройствам

Средние значения энергии на utterance и на секунду аудио:

- CPU:

  - `wav2vec2-base-960h`: ≈ 6.5 Дж/utter, ≈ 0.46 Дж/сек аудио;
  - `whisper-tiny`: ≈ 5.7 Дж/utter, ≈ 0.50 Дж/сек;
  - `whisper-base`: ≈ 9.9 Дж/utter, ≈ 0.87 Дж/сек;
  - `whisper-small`: ≈ 24.6 Дж/utter, ≈ 1.95 Дж/сек;
  - `whisper-large-v3`: ≈ 44.1 Дж/utter, ≈ 4.19 Дж/сек;
  - `whisper-medium`: ≈ 63.9 Дж/utter, ≈ 5.24 Дж/сек.

- GPU:
  - `wav2vec2-base-960h`: ≈ 21.8 Дж/utter, ≈ 1.46 Дж/сек;
  - `whisper-tiny`: ≈ 36.0 Дж/utter, ≈ 3.04 Дж/сек;
  - `whisper-base`: ≈ 43.9 Дж/utter, ≈ 3.39 Дж/сек;
  - `whisper-small`: ≈ 63.0 Дж/utter, ≈ 4.60 Дж/сек;
  - `whisper-large-v3`: ≈ 53.1 Дж/utter, ≈ 4.55 Дж/сек;
  - `whisper-medium`: ≈ 118.5 Дж/utter, ≈ 8.56 Дж/сек.

Отношение энергия(GPU)/энергия(CPU) на utterance:

- `wav2vec2-base-960h`: ≈ 3.3×;
- `whisper-tiny`: ≈ 6.3×;
- `whisper-base`: ≈ 4.4×;
- `whisper-small`: ≈ 2.6×;
- `whisper-large-v3`: ≈ 1.2×;
- `whisper-medium`: ≈ 1.9×.

То есть при выбранной модели мощности GPU:

- для лёгких моделей (tiny/base/wav2vec2) CPU оказывается более энергоэффективным в пересчёте на utterance;
- для тяжёлых моделей (large-v3, medium) прирост энергии при переходе на GPU относительно небольшой, при этом выигрыш по времени — очень большой.

Зависимости энергии от длины аудио показаны на рис. 6 и 7.

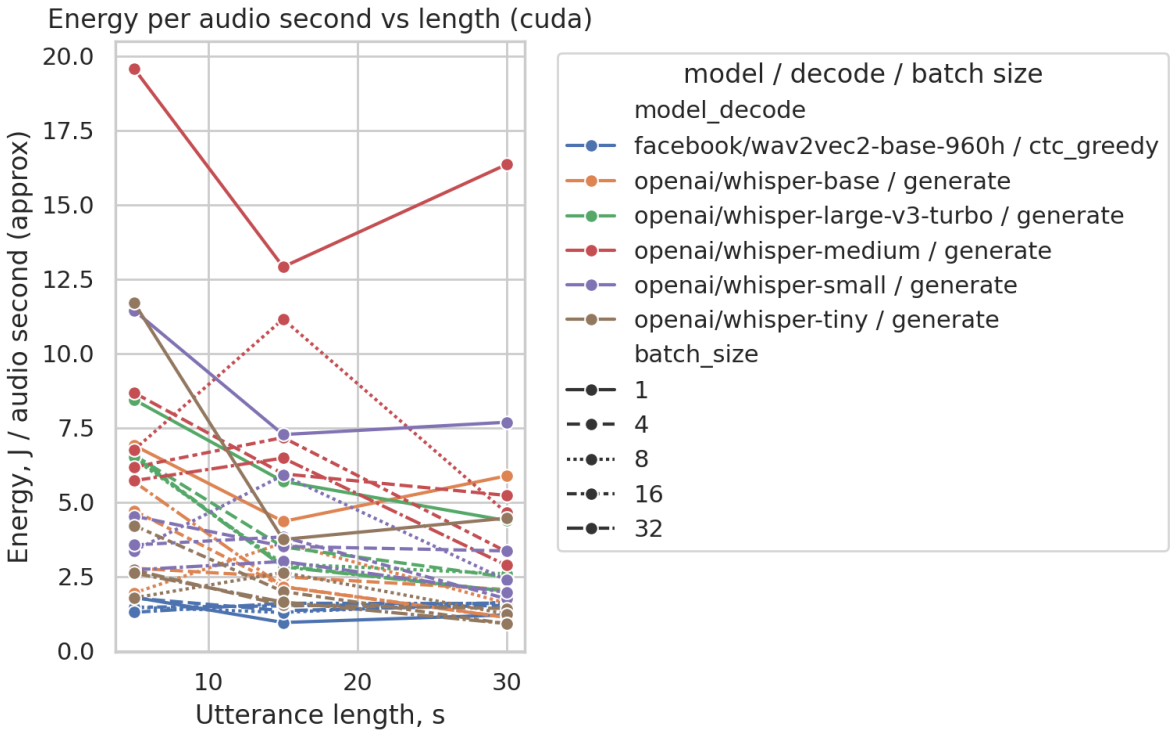


Рис. 6: Энергия на секунду аудио vs длина (device = cuda).

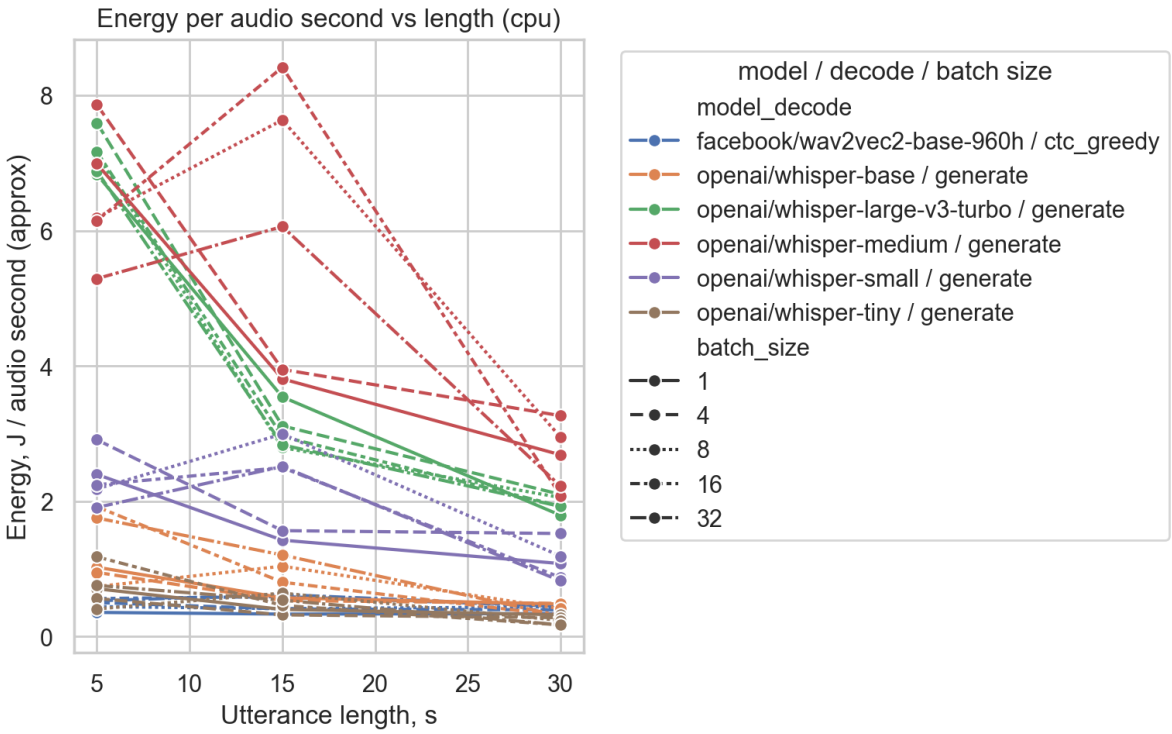


Рис. 7: Энергия на секунду аудио vs длина (device = cpu).

Компромисс энергия/latency для GPU показан на рис. 8.

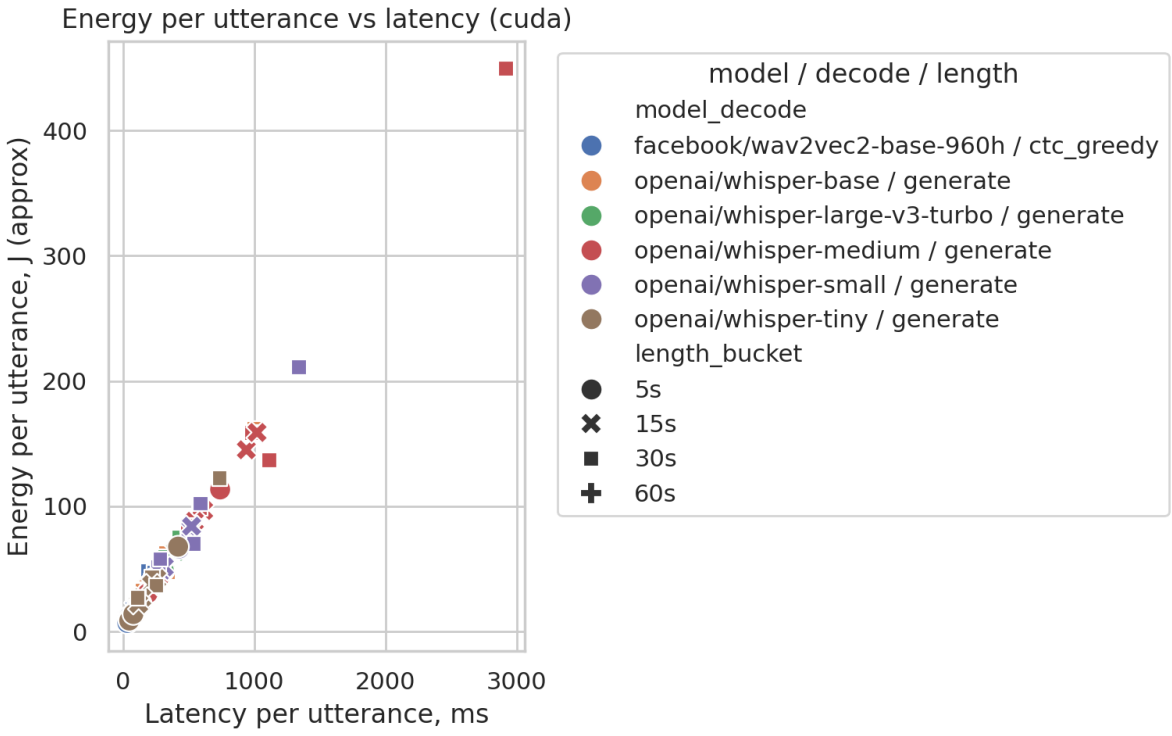
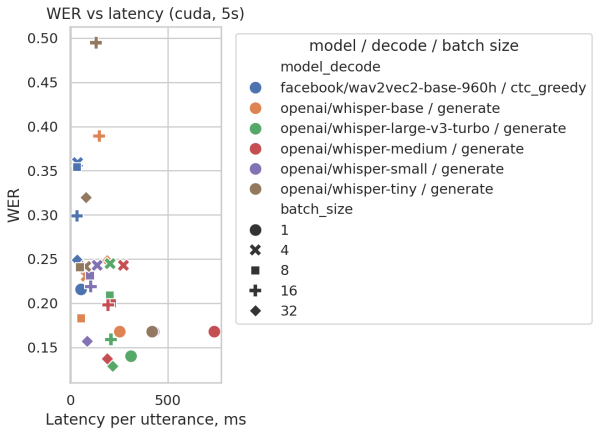


Рис. 8: Энергия на одно высказывание vs latency (device = cuda).

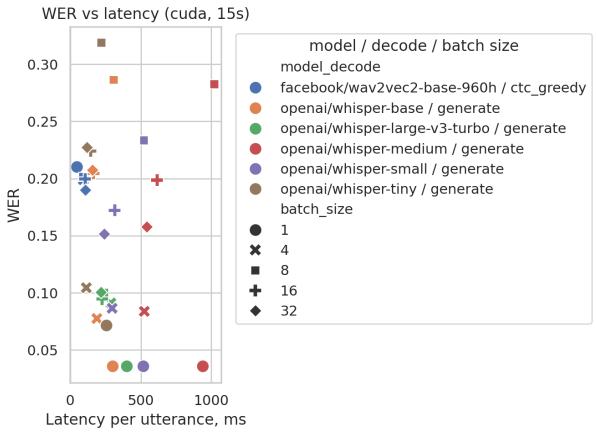
7 Trade-off скорость / качество / FLOPs

7.1 WER vs latency

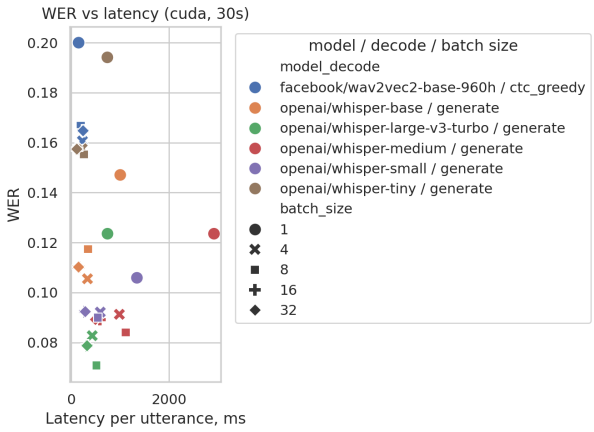
Для GPU-графика trade-off “WER vs latency” по разным бакетам длины показаны на рис. 9.



(a) Бакет 5 s



(b) Бакет 15 s



(c) Бакет 30 s

Рис. 9: WER vs latency (device = cuda) для разных бакетов длины.

Наблюдения по GPU:

- в “левом верхнем углу” (минимальная latency, максимальный WER) — лёгкие модели:
  - `whisper-tiny`,
  - `wav2vec2-base-960h`;
- `whisper-small` и `whisper-medium` лежат ближе к “середине”: существенно лучшее качество и приемлемая latency;
- `whisper-large-v3-turbo` обеспечивает минимальный WER, но имеет большую задержку (особенно на длинных аудио), хотя на GPU она всё ещё вменяемая (сотни миллисекунд).

7.2 WER vs RTF

Компромисс между качеством и real-time factor показан на рис. 5 (см. выше). Он позволяет видеть, какие модели дают RTF < 1 (работа быстрее реального времени) при заданном WER.

7.3 Latency vs FLOPs

График зависимости latency от GFLOPs на GPU приведён на рис. 10.

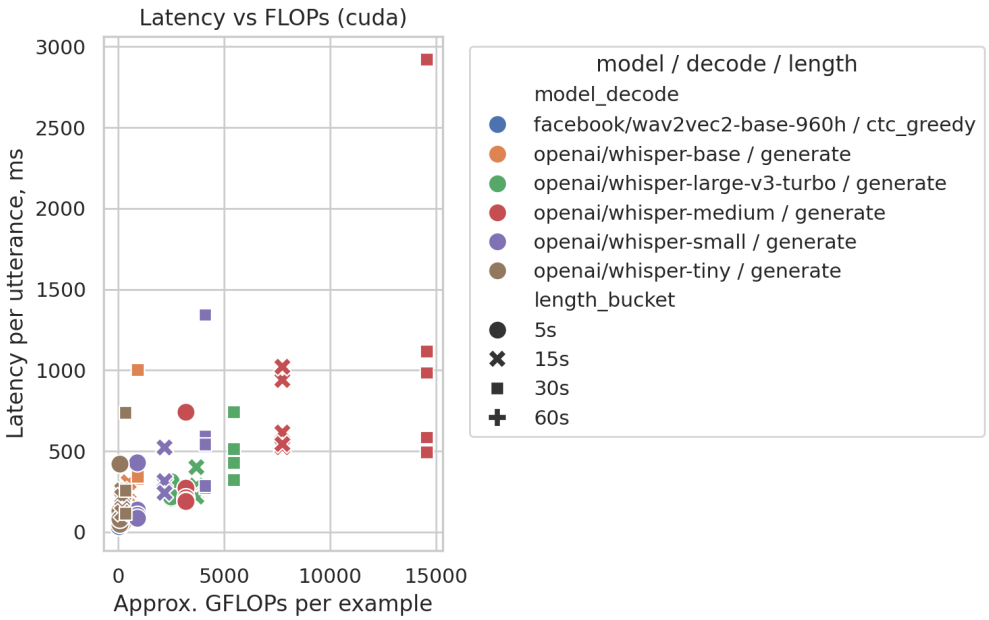


Рис. 10: Latency vs FLOPs (device = cuda).

В целом:

- наблюдается ожидаемый рост latency с увеличением FLOPs;
- `wav2vec2-base-960h` имеет наименьшие FLOPs и одну из минимальных задержек;
- семейство Whisper демонстрирует рост latency при переходе tiny → base → small → medium → large-v3, что хорошо коррелирует с GFLOPs;
- разброс точек показывает, что помимо FLOPs существенен вклад декодера и особенности реализации (особенно для авторегрессионных моделей).

8 Использование памяти GPU

Зависимость максимальной использованной видеопамяти от batch size показана на рис. 11.

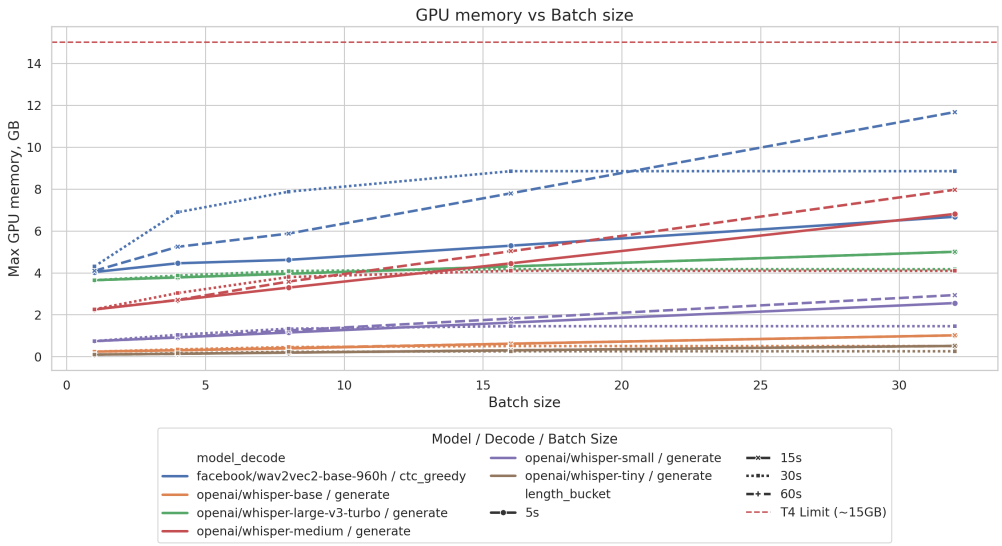


Рис. 11: Максимальная видеопамять vs batch size (device = cuda).

Выводы:

- использование памяти растёт с batch size почти линейно;
- крупные модели (`whisper-medium`, `whisper-large-v3-turbo`) быстрее всего упираются в 16 ГБ T4;
- для практических задач разумно выбирать умеренный batch size для крупных моделей, чтобы не получить OOM и не ухудшать latency из-за конкуренции за память.

9 Разбиение latency по стадиям (узкие места)

9.1 Breakdown на GPU и CPU

Разбиение latency по стадиям (preproc, encoder, decoder/CTC) для `batch_size = 1` показано на рис. 12 и 13.

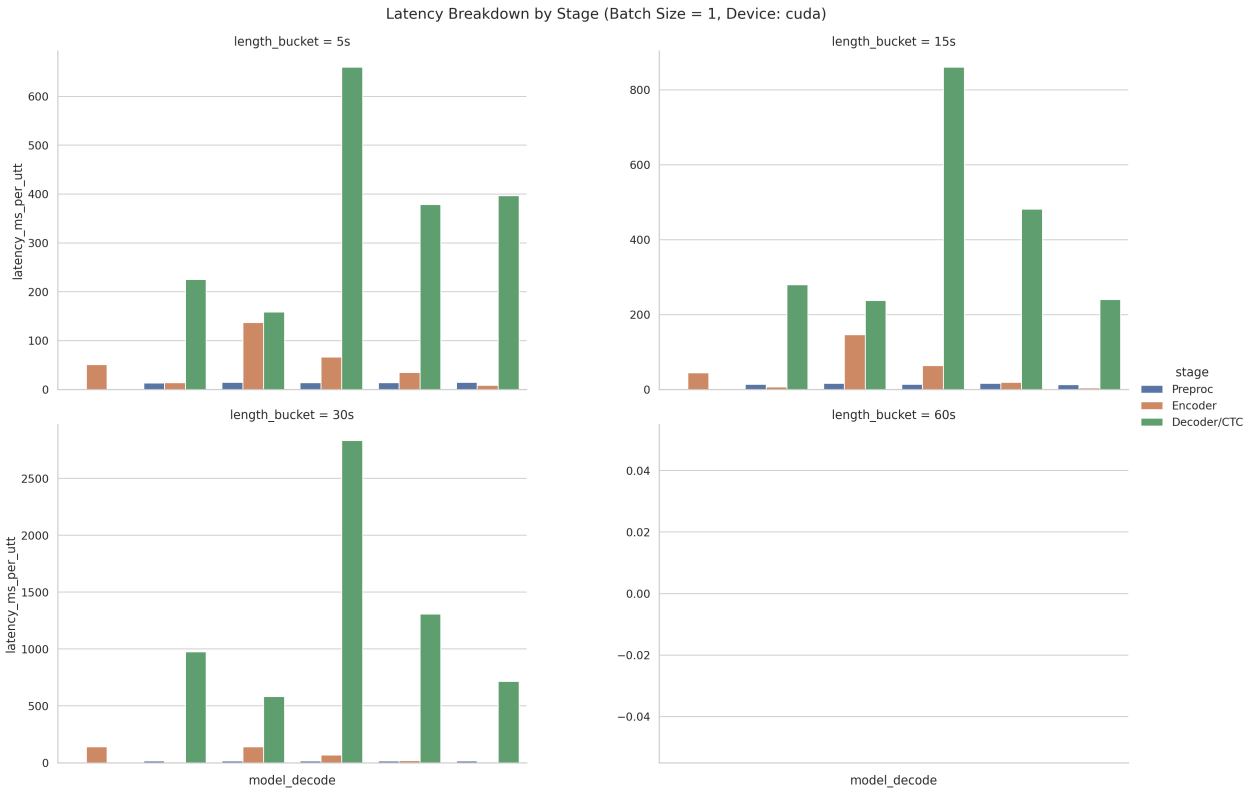


Рис. 12: Разбиение latency по стадиям (batch_size = 1, device = cuda).

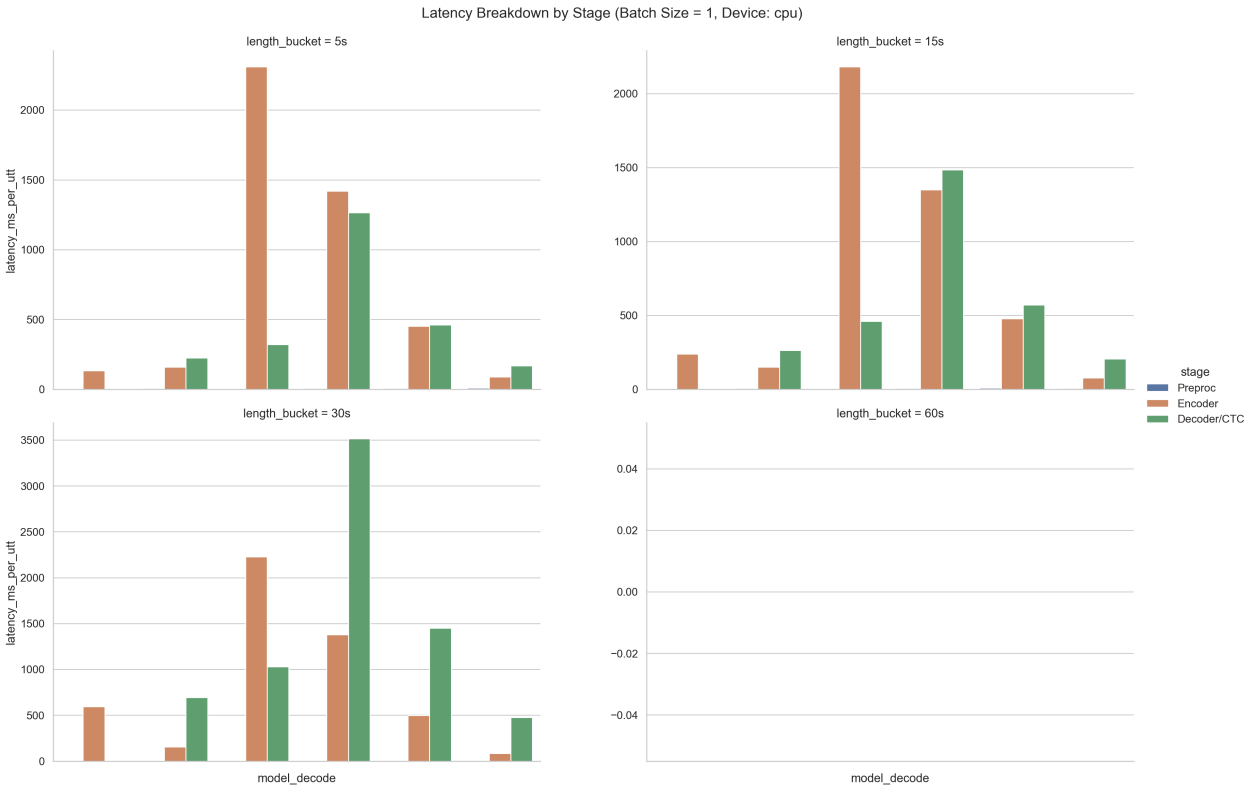


Рис. 13: Разбиение latency по стадиям (batch_size = 1, device = cpu).

Помимо сами графиков, по средним величинам (по всем конфигурациям) получаются следующие доли времени (в процентах, суммарно $\approx 100\%$):

CPU:

- `facebook/wav2vec2-base-960h`:
  - препроцессинг $\approx 0.1\%$,
  - энкодер $\approx 99.7\%$,
  - CTC-декодер $\approx 0.1\%$.
- `openai/whisper-tiny/base/small/medium`:
  - препроцессинг от 0.1 до 1%;
  - энкодер $\approx$ 20–32%;
  - декодер $\approx$ 68–79%.
- `openai/whisper-large-v3-turbo`:
  - препроцессинг $\approx 0.1\%$;
  - энкодер $\approx 81\%$;
  - декодер $\approx 18\%$.

GPU (T4):

- `facebook/wav2vec2-base-960h`:
  - препроцессинг $\approx 2\%$;
  - энкодер $\approx 96\%$;
  - CTC-декодер $\approx 2\%$.
- `openai/whisper-tiny/base/small/medium`:
  - препроцессинг от 3 до 10%;
  - энкодер $\approx$ 2–11%;
  - декодер $\approx$ 86–88%.
- `openai/whisper-large-v3-turbo`:
  - препроцессинг $\approx 8\%$;
  - энкодер $\approx 50\%$;
  - декодер $\approx 42\%$.

Итак:

- для **wav2vec2-base-960h** узкое место практически целиком в энкодере (на CPU и особенно на GPU);
- для **Whisper**:
  - на CPU для medium/small/tiny/base доминирует декодер (70% и более), но вклад энкодера тоже заметен;
  - на GPU для тех же моделей основное время уходит в авторегрессионный декодер (до 86–88%); энкодер становится относительно дешёвым;
  - для `whisper-large-v3-turbo` на GPU время делится примерно пополам между энкодером и декодером.
- препроцессинг на обоих устройствах даёт небольшой вклад и не является главным узким местом.

10 Сводные рекомендации

10.1 Выбор модели под сценарий

Real-time, короткие команды, ограниченные ресурсы.

- Устройства: CPU или слабый GPU.
- Модели: `whisper-tiny`, `whisper-base`, `wav2vec2-base-960h`.

- Плюсы:
  - минимальная latency;
  - низкое потребление памяти;
  - простота развёртывания на CPU.
- Минусы:
  - WER 18–23%, что заметно хуже `whisper-small/medium/large-v3`.

Онлайн-конференции, стримы (баланс скорость/качество).

- Устройства: GPU (T4 или лучше).
- Рекомендуемая модель: `whisper-small`.
- Аргументы:
  - WER $\approx 14.8\%$;
  - средняя latency на GPU $\approx$ 393 ms/utter, RTF ≈ 0.029 (быстрее реального времени);
  - разумное энергопотребление и использование памяти.

Оффлайн транскрибация (лекции, подкасты) с упором на качество.

- Устройства: GPU.
- Модели: `whisper-large-v3-turbo` и `whisper-medium`.
- Плюсы:
  - наилучший WER (large-v3 $\approx 11.9\%$);
  - GPU даёт ускорение 5–9$\times$ по сравнению с CPU.
- Минусы:
  - повышенное энергопотребление;
  - большие требования к видеопамяти.

10.2 Оптимизация времени и энергии

Исходя из анализа узких мест:

- **Encoder-first оптимизация**:
  - критичен для CTC-моделей (`wav2vec2`);
  - частично критичен для крупных моделей Whisper на CPU;

  - подходы: квантование (int8/fp16), pruning, distillation, использование ONNX Runtime / TensorRT.

- **Оптимизация декодера (Whisper)**:
  - на GPU именно декодер становится доминирующим узким местом для tiny/base/small/medi
  - возможные меры:
    * уменьшение `max_new_tokens`, если нет длинных текстов;
    * уменьшение beam size;
    * ранняя остановка на основе confidence;
    * более простые стратегии декодирования для менее критичных сценариев.

- **Batching и планирование**:
  - на GPU имеет смысл использовать умеренные batch size (4–16), особенно в оффлайн-сценариях, повышая throughput и энергоэффективность;
  - на CPU рост batch size выше 4 обычно даёт небольшой выигрыш и может ухудшать latency из-за ограничений памяти.

11 Ограничения и дальнейшая работа

Основные ограничения работы:

- Используется один датасет (TEDLIUM dev), англоязычный, с лекционным стилем речи; результаты могут отличаться на других языках и доменах (шумы, диалекты, спонтанная речь).

- Рассматриваются только два типа устройств: CPU и GPU T4; на других GPU (A100, L4, RTX) и на мобильных платформах картина может отличаться.

- Энергопотребление оценивается по простой модели (мощность $\times$ время), а не измеряется аппаратно (например, по `nvidia-smi` или внешним измерителям).

- FLOPs оцениваются по профайлерам (`thop`, `fvcore`) и не учитывают всех аспектов реальной производительности (кэш, пропускная способность памяти и др.).

Направления дальнейшей работы:

- добавить другие датасеты (шумные, многоканальные записи, другие языки);

- использовать более точные метрики энергии (логирование `nvidia-smi`, внешние ваттметры);

- исследовать влияние квантования, pruning и distillation на WER, latency и энергопотребление;

- сравнить с другими архитектурами (Conformer, более современные CTC/Transducer-модели).

12 Заключение

В работе проведено профилирование шести ASR-моделей на датасете TEDLIUM Release 3 (dev set) на CPU и GPU T4. На основе собранных CSV (`asr_profiling_cpu_only.csv`, `asr_profiling_gpu_only.csv`) были проанализированы:

- качество распознавания (WER/CER),
- латентность и RTF,
- энергопотребление,
- FLOPs,
- использование видеопамяти и разбиение времени по стадиям.

Основные выводы:

- Лучшая модель по качеству — `whisper-large-v3-turbo` (WER $\approx 11.9\%$); `whisper-small` и `whisper-medium` дают WER около 14.8% при меньших ресурсах.
- GPU T4 ускоряет inference по сравнению с CPU от $1.8\times$ до $9\times$ по латентности, особенно выигрывают крупные модели (`whisper-large-v3-turbo`, `whisper-medium`).
- Энергопотребление на GPU выше, чем на CPU, однако для самых тяжёлых моделей рост энергии умеренный (1.2–$1.9\times$) при большом выигрыше по времени.
- Для `wav2vec2` основное узкое место — энкодер; для семейства Whisper:
  - на CPU значимый вклад дают и энкодер, и декодер (особенно на длинных аудио);
  - на GPU для большинства размеров (tiny/base/small/medium) доминирует авторегрессионный декодер.
- В качестве универсального компромисса по “скорость/качество/ресурсы” для GPU можно рекомендовать `whisper-small`; для максимального качества в оффлайн-сценариях — `whisper-large-v3-turbo`.

Таким образом, цель проекта достигнута: на основе реальных измерений показано, как масштабируются ASR-модели по длине аудио и размеру батча на CPU и GPU, какие модели выгодны в разных сценариях и какие части вычислительного графа требуют первоочерёдной оптимизации с точки зрения латентности и энергопотребления.