Optimizing Whisper: Accelerating Runtime for Efficient Speech Processing

Alex Darwiche, Kavya Kathiravan, William Holden

UNIVERSITY OF GEORGIA

Agenda

Optimizing Whisper

Developed by OpenAI, Whisper is an **automatic speech recognition (ASR)** model that takes in audio files and can generate a **textual representation**. The model was trained on 680,000 hours of labeled audio data using "weakly supervised training" data.

Whisper uses an encoder/decoder Transformer-based architecture.

- **Audio-Encoder:** Converts speech into an intermediate representation.
- **Text-Decoder:** Translates the representation into text.

Speech-to-text applications are important for accessibility, media, and real-time transcription.

- Quickly transcribing large audio files can be time and memory expensive
- Speech-to-text has many accessibility benefits especially for hearing impaired persons

Motivation

Given the many applications of Whisper, it is of the utmost importance that it can run quickly in the real world. Improving the **inference speed** & **memory usage** of Whisper will allow deployment of the model to **edge devices** that have relatively small amounts of compute and memory.

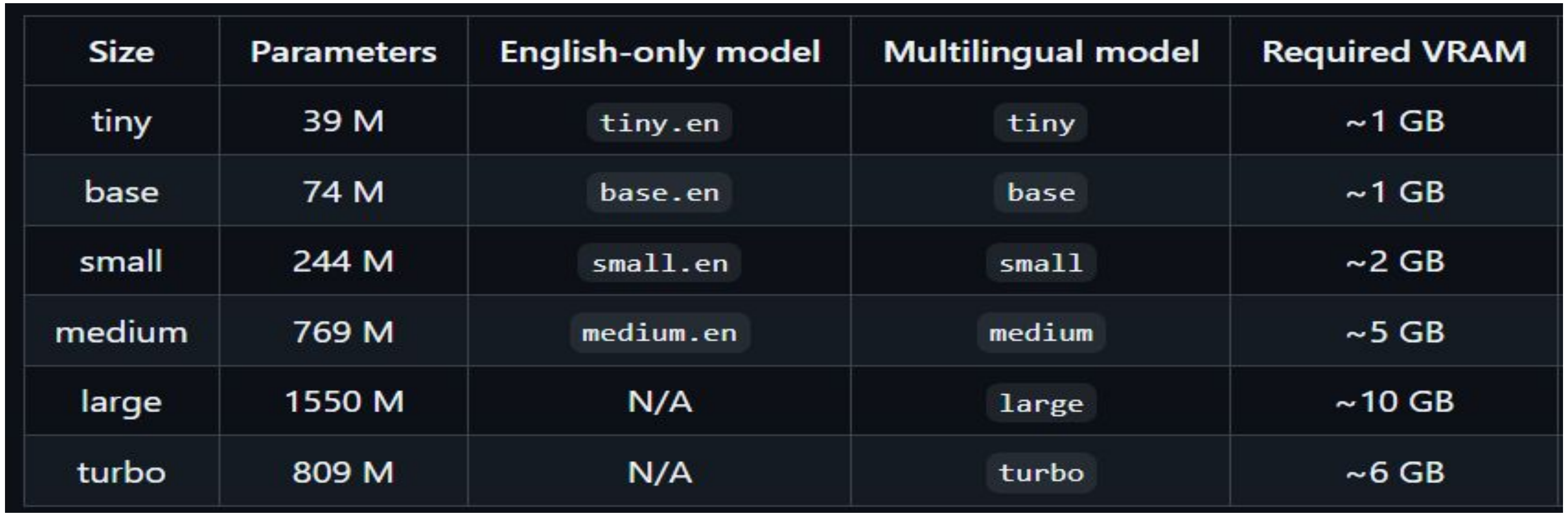
Key Problems:

1. **Speed:** Inference for Whisper is accurate, like human professional, but might need significant speed up to be deployed seamlessly on edge devices.
2. **Memory**: Whisper requires about **~1GB of VRAM for Inference** even with the smallest models**.**

Current Whisper Speed/Memory Usage

Metrics Measured:

- **Inference time** (speed of transcription).
- **Memory consumption** (RAM used during inference).
- **Accuracy** is measured using **Word Error Rate (WER) and Character Error Rate (CER)** via the **JIWER** package.



Size	Parameters	English-only model	Multilingual model	Required VRAM
tiny	39 M	`tiny.en`	`tiny`	~1 GB
base	74 M	`base.en`	`base`	~1 GB
small	244 M	`small.en`	`small`	~2 GB
medium	769 M	`medium.en`	`medium`	~5 GB
large	1550 M	N/A	`large`	~10 GB
turbo	809 M	N/A	`turbo`	~6 GB

High Level Project Plan

Improve Inference Speed:
Implement techniques learned in class - model quantization and better memory management, to improve inference speed.

AND…

Reduce Memory Usage for Inference:
Optimize memory usage to run the model on edge devices.

Techniques:
- **Quantization: FP32 to FP16 to INT8**
- **Winograd replacement for Convolutional Layers**
- Model Distillation (Stretch Goal)
- Flash Attention (Stretch Goal)

Milestones and Timeline

Expected Milestones:

- **March 10th -** Solidify Benchmark/Baseline speed and memory metrics
- **March 17th -** Implement Quantization
- **March 24th -** Implement Winograd for convolutional layers of Whisper
- **March 31st -** First Functional Model Distillation/Flash Attention Implementation
- **April 7th -** Begin collection and comparison of testing outputs

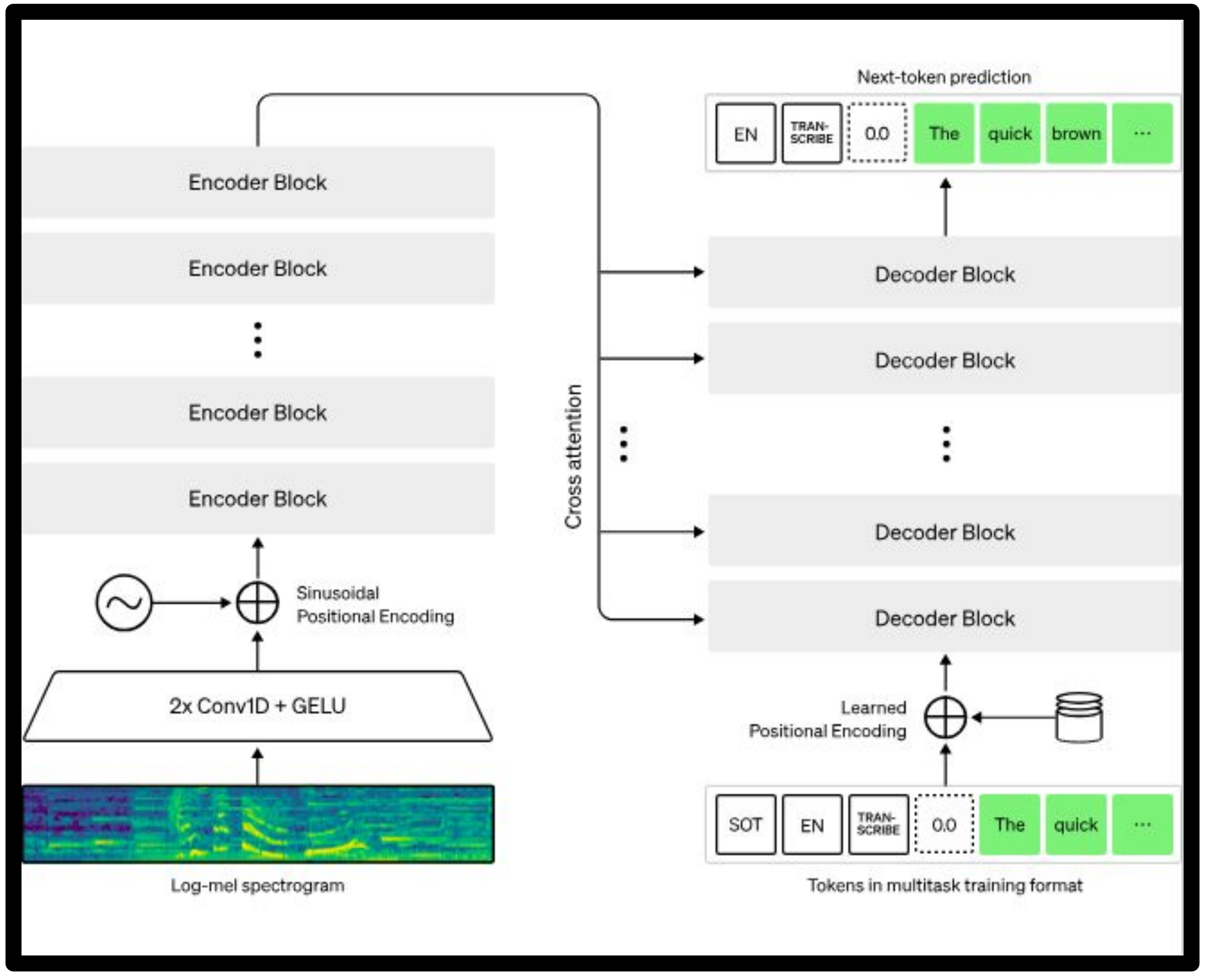
Techniques

Quantization:

- Plan to test the effects on model size and inference speed when moving from FP32 to FP16.

Winograd:

- Plan to implement Winograd-1D on the Conv1D layers in the audio encoding portion of the Whisper model to optimize inference speed and reduce VRAM usage.



Expected Outcomes

Looking to reduce the RAM needed to run an inference with the Base-English Whisper model, while maintaining similar levels of accuracy in terms of Word Error Rate (WER).

- We hope to reduce the RAM needed for inference for the base model, ideally staying below ~1GB used. This will allow the base model to be deployed on edge devices like smartphones.

Also plan to improve the speed of inference while similarly maintaining the levels of accuracy in the base model.

Additional Considerations

Whisper is trained mostly on English language audio, though it has the ability to translate and transcribe many other languages. Whisper's accuracy however, is much worse in other languages. With this in mind, we plan to implement our optimizations first on the **English-Only** model.

The Whisper engineers make they make the assertion that the model cannot be used for "**Real-Time**" transcription tasks. We believe that our series of optimizations will move Whisper closer to real-time tasks.

CPU vs GPU testing: Planning to do our initial benchmarking and testing with both CPUs and GPUs through Google Colab.

Resources

- Whisper Overview (https://github.com/openai/whisper/blob/main/model-card.md)
- Flash Attention (https://arxiv.org/pdf/2205.14135) - Tiling and Recomputation
- Winograd (https://arxiv.org/pdf/2201.10369)
- Quantization (https://pytorch.org/docs/stable/quantization.html)
- Whisper Paper (https://arxiv.org/pdf/2212.04356)

Thank You!

Current Performance and Challenges

Key Points:

- **Whisper offers high accuracy**, but:
    - **Slow inference speed** affects real-time usability.
    - **Large memory consumption** prevents deployment on smaller devices.
- **Example Benchmark Results (Base Model, FP32):**
    - CPU inference: **X seconds per audio clip**
    - GPU inference: **Y seconds per audio clip**
    - RAM usage: **Z MB**
- Our goal is to **optimize Whisper's efficiency without significantly sacrificing accuracy**.

Optimization Strategies

Key Points:

- **Step 1:** Benchmark Whisper’s performance (Baseline metrics).
- **Step 2:** Apply optimizations:
  - **Model Quantization** (FP16, INT8)
  - **Parallelization & Hardware Utilization**
  - **Memory Optimization & Efficient Data Loading**
  - **Model Pruning & Compression**
- **Step 3:** Compare and evaluate speed, memory, and accuracy trade-offs.

Model Quantization – Reducing Precision for Faster Inference

Key Points:

- Quantization **reduces precision (FP32 → FP16 or INT8)** to improve speed and efficiency.
- **Why it works:** Lower precision requires **fewer calculations and less memory**.
- **Implementation:**
  - Convert the Whisper model to **FP16 or INT8** using PyTorch.
  - Test for accuracy degradation and performance improvements.
- **Expected Impact:** Faster inference, lower memory consumption.

Memory Optimization & Pruning

Key Points:

- Reduce RAM consumption to make Whisper viable for **edge devices**.
- **Techniques:**
  - Optimize memory allocation for large tensors.
  - Implement **pruning** to remove unnecessary parameters.
  - Reduce **batch size dynamically** based on available memory.
- **Expected Impact:**
  - Whisper runs on **devices with lower RAM**.
  - More efficient inference on **mobile and embedded systems**.

Benchmarking and Evaluating Optimizations

Key Points:

- **Performance Metrics We Evaluate:**
  - **Inference Speed** – How much faster is the optimized model?
  - **Memory Usage** – How much RAM is required for inference?
  - **Accuracy Impact** – Compare WER/CER before and after optimization.
- **Visualization of Results (Graph/Table Comparison).**
- **Final Goal:** Achieve significant speedup **without major accuracy loss**.