**Georg-August-Universität Göttingen**

MSc. Data Science

# Performance Benchmarking and Evaluation of Open-Source ASR Frameworks for Real-Time Transcription

|  |  |
|---|---|
| **Submitted by:** | Deepak Budha |
| **Direct Supervisor:** | Nargex Lux |
| **Supervisor:** | Prof. Dr. Julian Kunke |
| **Submission Date:** | January 16, 2026 |

Faculty of Mathematics and Computer Science

Georg-August-Universität Göttingen

| | |
|---|---|
| **Family Name, Given/First Name** | Budha, Deepak |
| **Matriculation Number** | 14345584 |

## Acknowledgement

First and foremost, I would like to express my sincere gratitude to my supervisor, Professor Julian Kunkel, and to Dr. Narges Lux. Their invaluable guidance, support, and encouragement were essential throughout this project. Without their expertise and thoughtful input, the successful completion of this work would not have been possible.

## Abstract

Automatic Speech Recognition (ASR) systems are increasingly integral to modern applications, including voice assistants, transcription platforms, and accessibility tools. This research project focuses on the development of a real-time transcription service aimed at enhancing accessibility for individuals with hearing impairments. A primary use case is the transcription of spoken lectures in academic environments, enabling students with hearing difficulties to effectively follow and review course content.

The project evaluates three prominent open-source ASR models—Faster-Whisper, WhisperX, and Whisper.cpp, all utilizing the Large-v2 parameter set—by comparing their performance in terms of transcription accuracy, processing speed, hardware efficiency, and ease of integration. These models are benchmarked under varied conditions, including diverse English accents, background noise, and hardware constraints. The goal is to determine the most practical model for real-time deployment in low-resource environments. Through extensive empirical analysis and qualitative evaluation, this work outlines the trade-offs and optimal use cases for each model, offering actionable insights for future development of accessible transcription systems.

# Contents

## 1 Introduction

Automatic Speech Recognition (ASR) has become a cornerstone of modern human-computer interaction, enabling systems to transcribe spoken language into written text. This capability powers applications such as voice assistants, real-time captioning, and meeting transcription[1, 2]. ASR plays a particularly important role in making content accessible across domains like education, healthcare, and media.

Real-time transcription is crucial in dynamic environments such as academic lectures, video conferencing, and live events, where delays can hinder comprehension and communication[3]. For individuals with hearing impairments, ASR systems provide a pathway to inclusive access, especially in educational contexts where equal participation is essential[4].

Despite advancements, challenges remain in deploying ASR solutions in low-resource settings—where access to GPUs or cloud infrastructure is limited. Current state-of-the-art ASR solutions often fall short in these environments due to two primary factors: high computational costs and deployment complexity. Standard transformer-based models typically require high-VRAM NVIDIA GPUs, which are often unavailable in standard academic or mobile settings. Furthermore, the heavy dependency on complex Python environments and large containerized runtimes can hinder the seamless integration of ASR into lightweight, cross-platform applications. Optimized open-source implementations, including Faster-Whisper[5], WhisperX[6], and Whisper.cpp[7], address these barriers by enabling high-parameter models to run on consumer-grade CPUs and integrated GPUs. These models offer promising alternatives by balancing speed, accuracy, and efficiency.

This report presents a comparative evaluation of these models, focusing on transcription accuracy, latency, hardware usage, and ease of integration. The goal is to identify the most practical solution for real-time accessibility applications in constrained academic environments.

## 2 Objectives and Scope

This research aims to develop and evaluate a real-time transcription service by identifying the most effective open-source ASR model for production deployment. We conduct a comprehensive comparison of three leading ASR models—*Faster-Whisper*, *WhisperX*, and *Whisper.cpp*—across multiple performance and deployment metrics to guide optimal model selection.

The key objectives are to:

- Systematically evaluate the selected ASR models.
- Benchmark a range of metrics including:
  - Transcription accuracy (Word Error Rate, Character Error Rate)

  - Processing speed (Real-Time Factor)
  - Resource efficiency (Peak Memory Usage, Model Load Time)
  - Feature completeness (Word-level timestamps, Speaker diarization, Multi-language support)
  - Deployment readiness (Integration complexity, Quantization support, License compatibility)

- Test model performance on different audio conditions (clean, noisy, accented).
- Analyze trade-offs in performance and deployment aspects to inform production implementation.

## 2.1 Scope

While the selected models (Faster-Whisper, WhisperX, and Whisper.cpp) inherently support over 90 languages, this research focuses exclusively on English language transcription. This decision ensures a controlled environment for measuring Word Error Rate (WER) and latency without the variability introduced by differing linguistic complexities or dataset imbalances in non-English languages.

## 3 Overview of Selected Models

The evaluation focuses on three optimized versions of OpenAI's Whisper architecture. Each model was selected because it addresses specific limitations of the original PyTorch implementation, particularly regarding inference speed and deployment overhead.

### 3.1 Faster-Whisper

*Faster-Whisper* is a reimplementation of OpenAI's Whisper model using the `CTranslate2` inference engine. It utilizes **Float16 and INT8 quantization** to accelerate execution on both CPUs and GPUs. By using a specialized C++ backend for the transformer calculations while maintaining a Python interface, it achieves a significant speedup over the original model without a measurable loss in transcription accuracy.

### 3.2 WhisperX

*WhisperX* addresses the inherent "timestamp drift" found in standard Whisper outputs. It introduces a post-processing pipeline that uses phoneme-level forced alignment via the `Wav2Vec2` model. This ensures that the generated text is precisely synchronized with the audio signal, making it the ideal candidate for applications requiring high-precision captions.

### 3.3 Whisper.cpp

*Whisper.cpp* is a high-performance, native C/C++ port of the Whisper architecture with **zero Python dependencies**.

- **Architectural Contrast:** Unlike the other models, it is a standalone implementation that eliminates the overhead of the Python runtime and the heavy PyTorch ecosystem. This makes it highly portable and significantly easier to integrate into mobile or embedded systems.
- **Quantization Details:** For this study, the model was benchmarked using **4-bit integer quantization (Q4_0)**. This reduces the model size by approximately 75% compared to the original FP32 weights, allowing the *Large-v2* model to run comfortably within the 16GB unified memory of the MacBook Air M3 while maintaining competitive accuracy.

## 4 Methodology

The methodology focuses on evaluating the models through a comprehensive benchmarking framework designed to assess performance across multiple dimensions. This section details the experimental setup, dataset characteristics, and evaluation metrics used for comparative analysis.

### 4.1 Experimental Setup

The benchmarking was conducted on a **MacBook Air M3** with Apple Silicon architecture, providing a representative environment for modern edge computing scenarios. The hardware specifications include:

- **CPU**: Apple M3 chip, 8-core (4 performance cores + 4 efficiency cores)
- **Memory**: 16 GB unified memory architecture
- **Operating System**: macOS 15.6.1 (Sequoia)

The evaluation framework was implemented using Python, with each model tested under identical conditions to ensure fair comparison. All models were evaluated using their Large-v2 variants to maintain consistency. While Large-v3 is available, Large-v2 was selected for this benchmark due to documented stability issues in Large-v3, specifically regarding unconditional repetition loops during long-form transcription, which would compromise the reliability of a real-time service.

### 4.2 Data Collection and Audio Samples

To ensure a robust evaluation, a synthetic dataset was generated to simulate different acoustic environments and speech lengths. Unlike standard short-segment benchmarks, this study specifically evaluated performance on long-form audio to mimic academic lecture conditions. The dataset was generated using the `edge-tts` engine to produce high-quality baseline audio.

- **Total Samples**: 9 audio files, evenly distributed across three categories.
- **Audio Duration**: Samples were generated with progressive lengths of **1 minute, 2 minutes, and 3 minutes**.
- **Diversity**: The dataset includes three distinct categories:
  1. **Clean Audio:** High-quality synthesized speech with clear diction and no background interference. (Samples: *clean_1min.wav*, *clean_2min.wav*, *clean_3min.wav*)

2. **Noisy Audio:** The same speech samples overlaid with **white noise at 15dB SNR** (Signal-to-Noise Ratio). This tests the model's ability to filter out steady-state background hiss, common in poor recording environments. (Samples: *noisy_1min.wav*, etc.)
3. **Accented Speech:** Samples generated using diverse English text-to-speech voices to evaluate phonetic flexibility.
   - 1 Minute: **British English** (en-GB-SoniaNeural)
   - 2 Minutes: **Australian English** (en-AU-NatashaNeural)
   - 3 Minutes: **Indian English** (en-IN-NeerjaNeural)

- **Processing**: All audio files were standardized to **16kHz mono WAV** format to ensure compatibility across all inference engines without the need for runtime transcoding.

## 4.3 Evaluation Metrics

The models were evaluated using a comprehensive set of metrics designed to assess both accuracy and performance characteristics:

- **Word Error Rate (WER):** Calculated as the percentage of word-level transcription errors, measuring accuracy at the word level. Lower values indicate better performance.
- **Character Error Rate (CER):** Calculated as the percentage of character-level transcription errors, providing a finer-grained accuracy measurement. Lower values indicate better performance.
- **Real-Time Factor (RTF):** Calculated as the ratio of inference time to audio duration (RTF = inference_time / audio_duration). Values less than 1.0 indicate real-time capability, meaning the model can process audio faster than it plays. Lower RTF values indicate faster processing.
- **Features:** Evaluated across four key capabilities: (1) *word-level timestamps* for precise temporal alignment of transcribed words, (2) *speaker diarization* for identifying and separating different speakers, (3) *multi-language support* for handling multiple languages, and (4) *language translation* for cross-lingual speech-to-text capabilities. Each feature is assessed as a binary capability (supported/not supported).
- **Deployment Metrics:** Assessed across three dimensions: (1) *ease of integration* evaluating the complexity of incorporating the model into production systems, (2) *quantization support* indicating whether the model supports model compression techniques for reduced memory footprint and faster inference, and (3) *license* type determining usage rights and commercial deployment permissions.

Each model was tested on all 9 audio samples across the three categories, resulting in 27 total evaluation runs (3 models $\times$ 9 samples). The metrics were aggregated to provide average performance across audio types and overall performance summaries.

## 5 Results

## 5.1 Performance

### 5.1.1 Real-Time Factor (RTF)

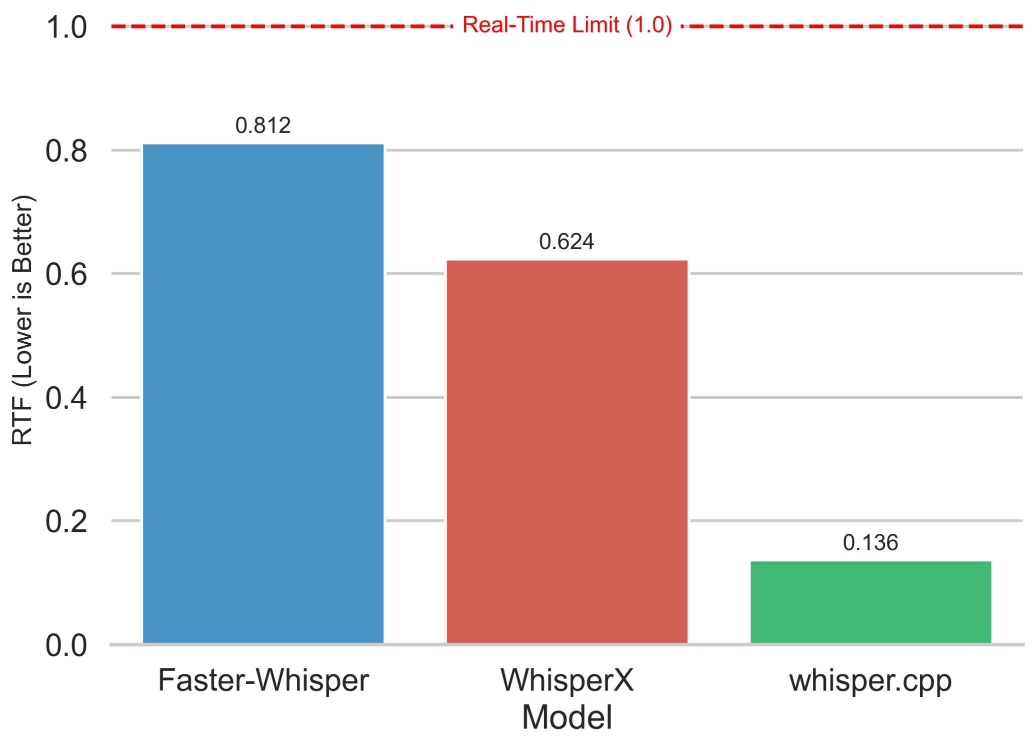


Figure 1: Average Real-Time Factor (RTF) comparison across models

Figure 1 shows the Real-Time Factor (RTF) for each model. RTF is the critical metric for live streaming; an RTF of 0.5 means the model can transcribe 1 hour of audio in 30 minutes (or 1 second of audio in 0.5 seconds).

- **Whisper.cpp** achieves exceptional performance with an RTF of **0.136**. This means it processes audio roughly **7x faster than real-time**, making it remarkably efficient for live streaming even on battery-powered devices.
- **WhisperX** demonstrates solid real-time capability with an RTF of **0.624**, comfortably staying under the 1.0 limit despite the overhead of its forced-alignment pipeline.
- **Faster-Whisper** shows an RTF of **0.812**. While still capable of real-time processing ($RTF < 1.0$), it is the slowest in this Apple Silicon environment due to the lack of MPS (Metal) support in its underlying CTranslate2 engine.

All three models are viable for real-time applications on the MacBook Air M3, but **Whisper.cpp** provides the largest safety margin, ensuring that the system can handle spikes in activity or background tasks without falling behind the live audio stream.

*Note: Faster-Whisper's performance was constrained by the lack of GPU (Metal) support in its underlying CTranslate2 engine on macOS. In a server environment equipped with NVIDIA GPUs (CUDA), Faster-Whisper is expected to significantly outperform the other models in throughput and latency, making it the preferred choice for cloud-based deployments.*

### 5.1.2 Word Error Rate (WER)

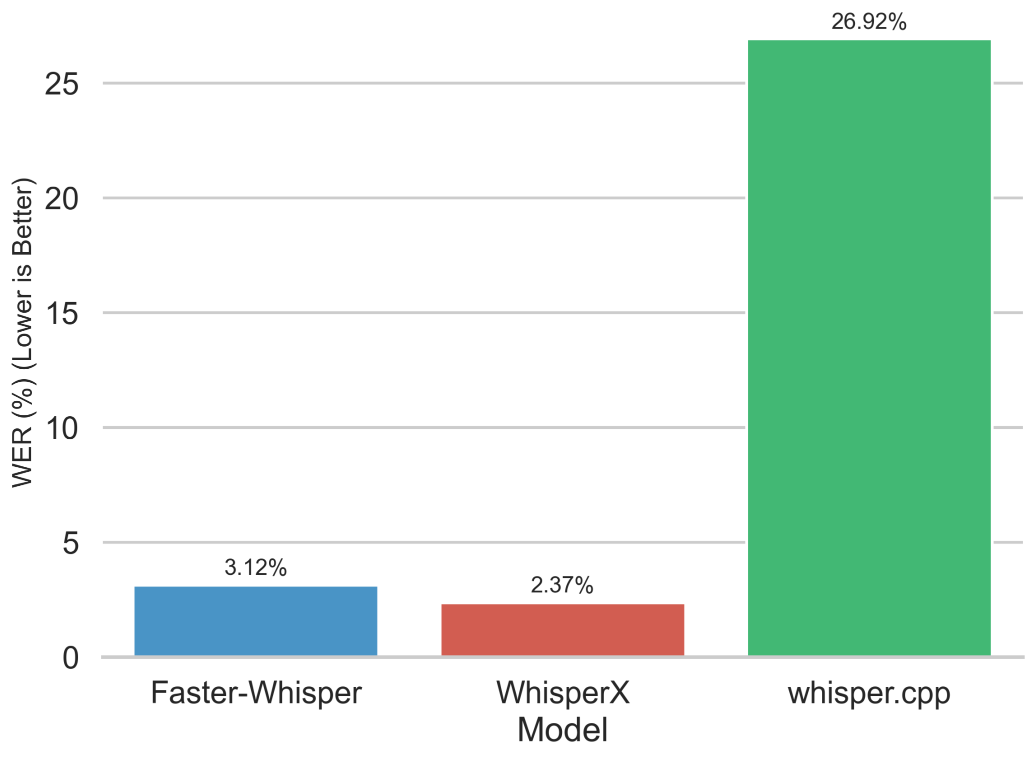


Figure 2: Average Word Error Rate (WER) comparison across models

Figure 2 presents the average Word Error Rate (WER) for each evaluated model, where lower values indicate superior transcription accuracy.

- **WhisperX** achieves the lowest WER at **2.37%**, representing the highest transcription accuracy among the tested systems. Its use of uncompressed FP16 weights combined with phoneme-level alignment contributes to highly precise outputs, making it particularly well-suited for offline or batch transcription scenarios where precision is the primary requirement.

- **Faster-Whisper** follows closely with a WER of **3.12%**. Despite employing optimized inference through CTranslate2, it maintains accuracy comparable to the original Whisper model, demonstrating that substantial inference-time optimizations can be achieved with minimal degradation in transcription quality.

- **Whisper.cpp** exhibits a substantially higher WER of **26.92%**. This performance gap is an expected consequence of its reliance on aggressive **4-bit integer quantization (Q4_0)**, which significantly reduces the memory footprint and improves inference speed at the cost of acoustic detail preservation. While the model generally captures the semantic content of speech, finer linguistic details are more frequently lost compared to full-precision implementations.

Overall, **WhisperX** emerges as the best-performing model for accuracy-critical applications, producing transcriptions that closely match human ground truth. **Faster-Whisper** offers a compelling balance between transcription accuracy and computational efficiency, making it a strong candidate for near-real-time systems. In contrast, **Whisper.cpp** prioritizes extreme resource efficiency and low latency, positioning it as a viable solution for deployment in memory-constrained or edge environments where approximate transcription is acceptable.

*Note: The elevated error rate (26.92% WER) observed in Whisper.cpp is primarily a result of the 4-bit integer quantization (Q4_0) used for this benchmark. This specific quantization level was selected to evaluate the model's performance under extreme resource constraints, simulating environments where minimizing memory footprint and battery consumption are critical. On systems with greater memory availability—allowing for 8-bit or 16-bit weights—the WER would likely converge toward that of the higher-precision models.*

### 5.1.3 Character Error Rate (CER)

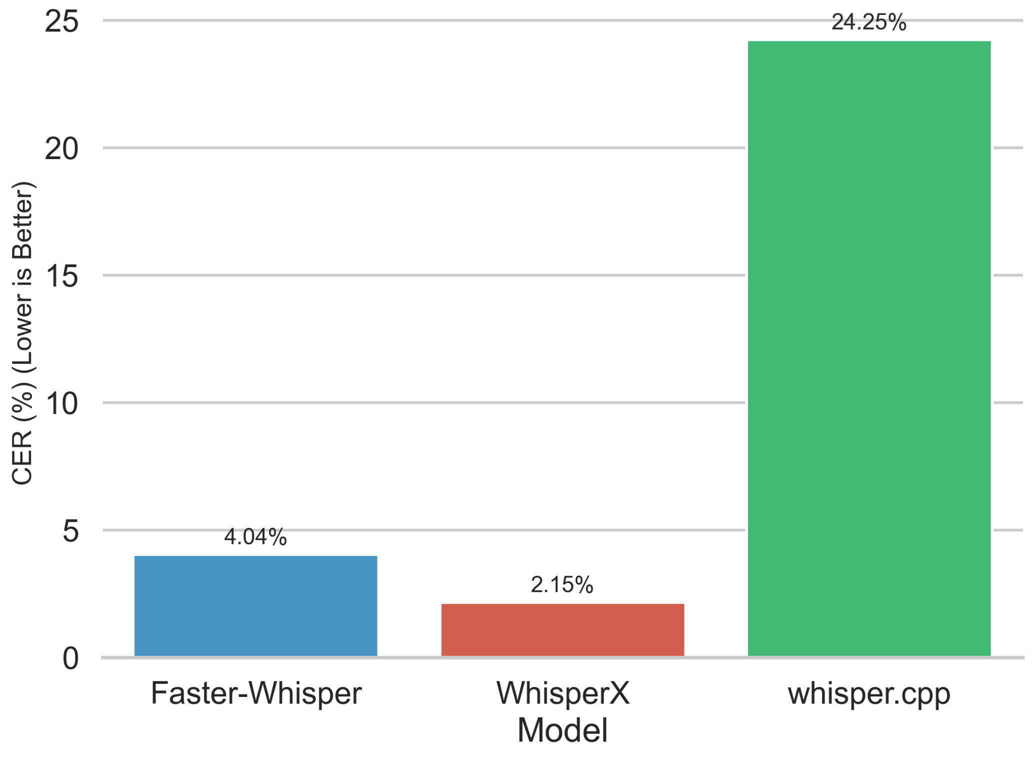


Figure 3: Average Character Error Rate (CER) comparison across models

Figure 3 shows the Character Error Rate (CER) for each evaluated model.

- **WhisperX** achieves the lowest CER at **2.15%**, confirming its superior fine-grained transcription accuracy.
- **Faster-Whisper** follows with a CER of **4.04%**, slightly higher than its WER (3.12%). This contrasts with the other models, where CER is typically lower than WER, highlighting subtle differences in character-level vs. word-level transcription performance.
- **Whisper.cpp** exhibits a high CER of **24.25%**, closely tracking its WER (26.92%). This indicates that aggressive **4-bit integer quantization (Q4_0)** affects both word recognition and character-level spelling consistency to a similar extent.
- The consistent performance gap between WhisperX/Faster-Whisper and Whisper.cpp reinforces that low-bit quantization significantly impairs the model's ability to preserve fine phonetic and orthographic details.

Overall, for applications requiring precise spelling—such as proper nouns or technical terminology—**WhisperX/Faster-Whisper** is the recommended choice. **Whisper.cpp**, while offering speed and efficiency, introduces a high rate of character-level deviations, which may necessitate substantial human post-editing.

### 5.1.4 Model Load Time

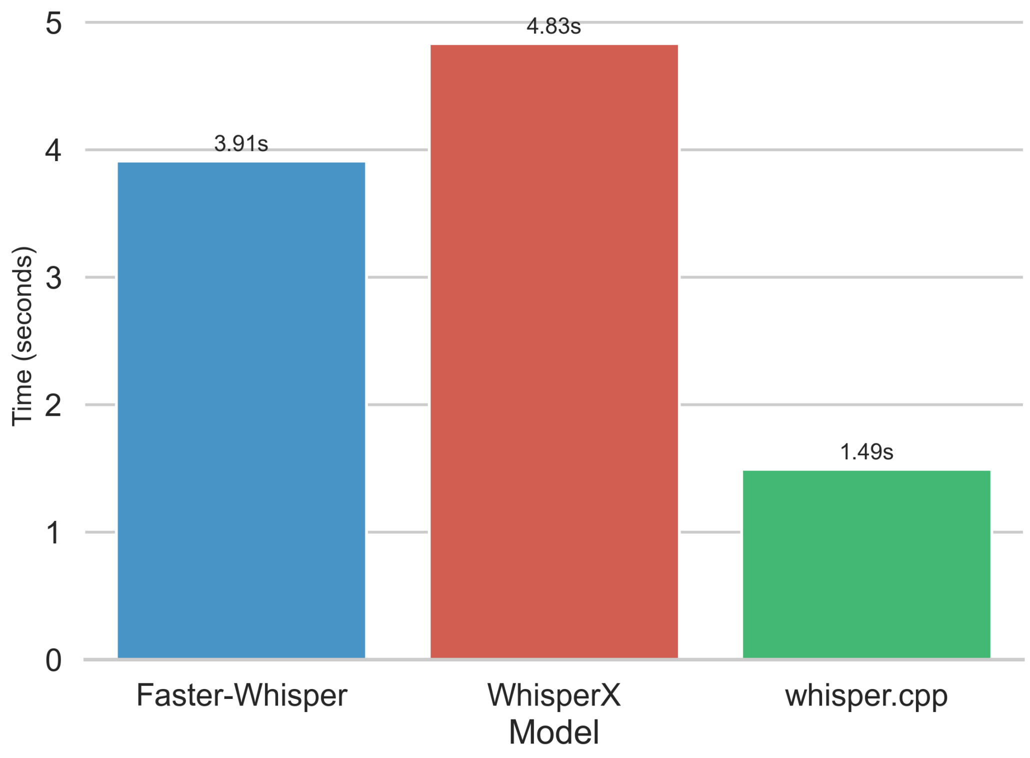


Figure 4: Average Model Load Time comparison across models

Figure 4 illustrates the initialization latency required to load weights into system memory.

- **Whisper.cpp** is the fastest to initialize at **1.49 seconds**. Its standalone C++ architecture eliminates the heavy overhead of the Python runtime and PyTorch framework, enabling near-instantaneous startup.
- **Faster-Whisper** shows a moderate load time of **3.91 seconds**. While optimized, it still incurs the "cold start" costs of initializing the Python environment and the CTranslate2 engine.
- **WhisperX** exhibits the longest load time at **4.83 seconds**. This is a direct consequence of its dual-model architecture, which requires loading both the primary Whisper weights and an auxiliary Wav2Vec2 model for phoneme alignment.

*Note: This metric is a primary differentiator for user-facing applications and CLI utilities where "Time to First Interaction" (TTFI) is a key performance indicator. However, for persistent server-side deployments or large-scale batch processing, these initialization costs are amortized over the total runtime and become a secondary concern.*

### 5.1.5 Peak Memory Usage

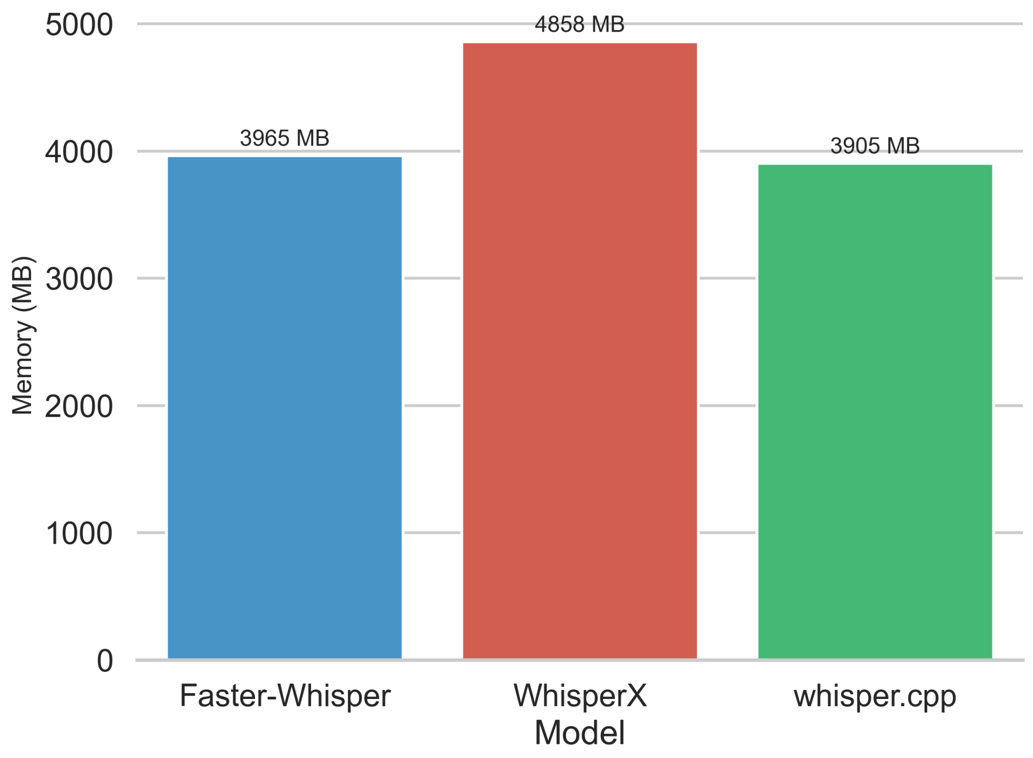


Figure 5: Average Peak Memory Usage comparison across models

Figure 5 illustrates the maximum Resident Set Size (RSS) memory consumed by each process during the transcription of a 3-minute audio sample.

- **WhisperX** has the highest memory footprint at approximately **4858 MB**. This significant overhead is attributed to the dual-model architecture, which holds both the primary Whisper weights and the auxiliary Wav2Vec2 model in memory, alongside the standard PyTorch framework requirements.
- **Faster-Whisper** shows efficient memory management at **3965 MB**. By leveraging the CTranslate2 engine, it bypasses the heavy autograd machinery of PyTorch, maintaining a leaner profile that is well-suited for deployment in Python-based microservices.
- **Whisper.cpp** achieves the lowest peak usage at **3905 MB**. This represents the most optimized footprint, consisting primarily of the static 4-bit quantized weights (≈1.5GB) and the dynamic buffers required for the audio context window.

For a production-grade transcription service, memory efficiency directly correlates with **concurrency** and **operational cost**. The ≈900 MB reduction offered by **Whisper.cpp** and **Faster-Whisper** allows for a higher density of simultaneous transcription streams on a single server instance. Furthermore, in edge-computing scenarios—such as a local service running on a MacBook Air—minimizing peak memory is vital to ensure the system remains responsive without triggering swap memory, which would otherwise introduce significant latency spikes in the live audio stream.

## 5.2 Features

### 5.2.1 Model Features Analysis

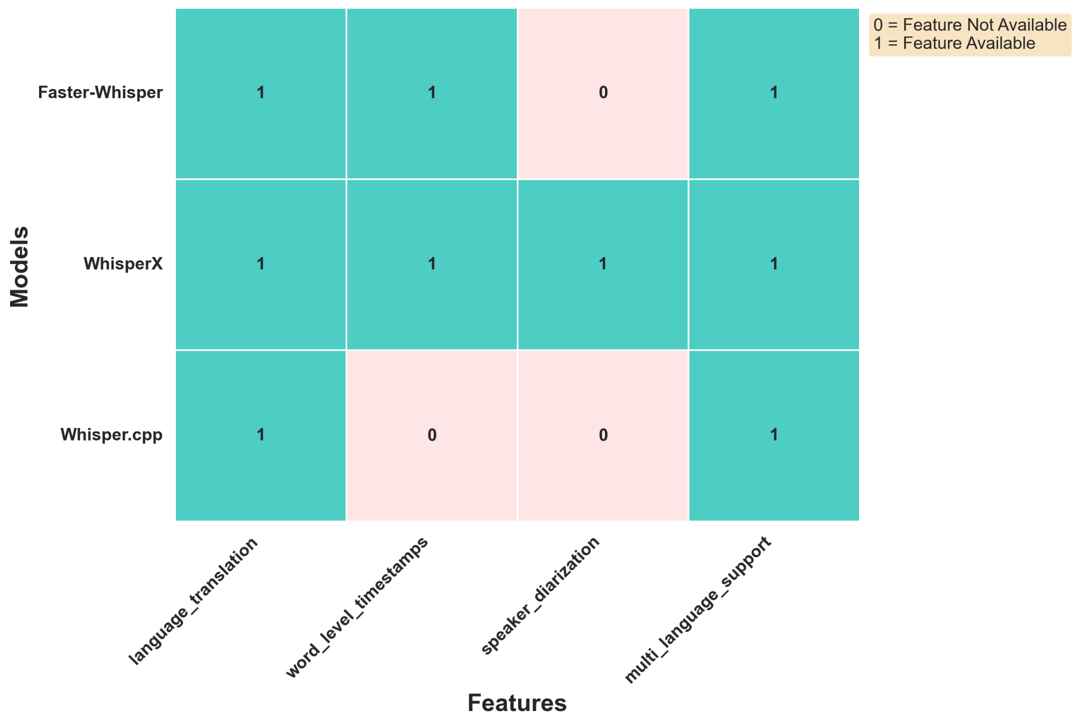


Figure 6: Feature comparison matrix across ASR models

Figure 6 shows the feature capabilities across Faster-Whisper, WhisperX, and Whisper.cpp:

- **Language Translation**: All three models support cross-language transcription (speech in one language to text in another), enabling multilingual workflows; Faster-Whisper and WhisperX inherit Whisper's translation capability, and Whisper.cpp provides translation via the ggml pipeline.
- **Word-Level Timestamps**: Faster-Whisper and WhisperX provide precise temporal alignment for subtitle generation and transcription synchronization, while Whisper.cpp lacks this capability.
- **Speaker Diarization**: Only WhisperX offers multi-speaker identification and separation, making it the exclusive choice for complex audio scenarios requiring speaker differentiation.
- **Multi-Language Support**: All three models provide comprehensive global language support, ensuring universal applicability across diverse linguistic contexts.

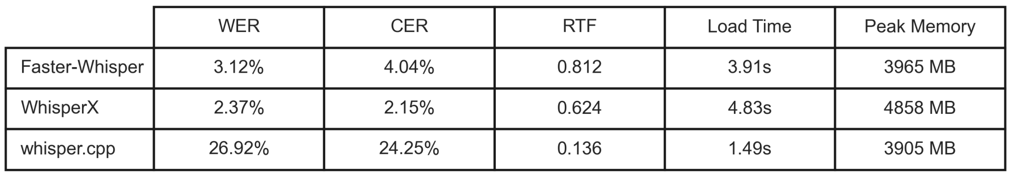
This analysis reveals that WhisperX offers the most feature-rich solution with exclusive speaker diarization capabilities, while Faster-Whisper provides a balanced feature set for most applications. Whisper.cpp, while efficient, has the most limited feature set, making it suitable primarily for basic transcription needs where advanced features are not required.

### 5.2.2 Deployment Metrics

- **Quantization Support**: Faster-Whisper supports quantization (e.g., 8-bit) via CTranslate2 to reduce model size and improve efficiency, and WhisperX inherits this capability through the Faster-Whisper backend. Whisper.cpp also supports native integer quantization (multiple bit-width formats such as Q4 and Q5) to reduce memory and disk footprint, improving deployment flexibility.
- **Ease of Integration**: Faster-Whisper offers seamless integration with developer-friendly APIs and tooling. WhisperX has moderate integration complexity due to its advanced features, while Whisper.cpp requires more effort to integrate because of its minimalistic implementation.
- **License**: All three models use the MIT License, providing maximum flexibility for commercial and non-commercial use without legal barriers.

This analysis demonstrates that Faster-Whisper offers the most developer-friendly deployment experience with excellent integration and quantization support, while WhisperX provides a balanced approach with slightly higher integration effort. Whisper.cpp, despite its efficiency, requires more development effort and lacks optimization features. The universal MIT licensing across all models ensures that legal considerations do not influence model selection, allowing focus on technical requirements and deployment constraints.

## 5.3 Overall Performance Assessment



| | WER | CER | RTF | Load Time | Peak Memory |
|---|---|---|---|---|---|
| Faster-Whisper | 3.12% | 4.04% | 0.812 | 3.91s | 3965 MB |
| WhisperX | 2.37% | 2.15% | 0.624 | 4.83s | 4858 MB |
| whisper.cpp | 26.92% | 24.25% | 0.136 | 1.49s | 3905 MB |

Figure 7: Comprehensive performance metrics summary across all models

Figure 7 illustrates a clear tri-modal distribution in performance characteristics across the evaluated frameworks. The quantitative data reveals three distinct optimization paths:

- **Accuracy Leadership: WhisperX** achieved the highest precision with a Word Error Rate (WER) of **2.37%**, though it required the highest memory overhead (**4858 MB**) and longest load times (**4.83s**).
- **Operational Balance: Faster-Whisper** maintained a balanced profile, delivering a competitive **3.12% WER** with significantly lower memory usage (**3965 MB**) compared to WhisperX.
- **Computational Speed: Whisper.cpp** proved to be the fastest by a wide margin with a Real-Time Factor (RTF) of **0.136**, processing audio roughly 7$\times$ faster than real-time. However, its 4-bit quantization resulted in a substantial accuracy trade-off, with WER rising to **26.92%**.

## 6 State of the Art in ASR

Automatic Speech Recognition (ASR) has seen rapid advances due to improvements in deep learning, transformer architectures, and large-scale pretraining. Several open-source and proprietary models now offer near-human-level transcription quality, especially for English and high-resource languages.

### 6.1 Recent Advances in ASR Models

**OpenAI Whisper**[8] marked a significant leap by introducing a multilingual, multitask transformer-based model trained on 680,000 hours of weakly supervised audio data. Whisper demonstrates strong generalization across languages, accents, and domains, with robust performance even on noisy inputs.

**NVIDIA NeMo**[9] is a flexible ASR toolkit offering state-of-the-art models like Conformer-CTC and Transducer, optimized for both training and deployment on GPUs. These models have been pretrained on massive datasets (e.g., NeMo's EN-US Conformer-Transducer on over 50k hours) and support both offline and streaming transcription.

**Other state-of-the-art models** include:

- **Wav2Vec 2.0** from Facebook AI, which uses contrastive learning for self-supervised representation learning from raw audio.
- **HuBERT** and **Conformer-based models**, which combine CNNs with transformers for better local and global feature modeling.
- **Whisper Large v3**, the largest Whisper model, provides significantly improved transcription accuracy and multilingual performance but requires considerable compute.

### 6.2 Proprietary APIs vs. Open-Source Models

Commercial ASR APIs such as Google Speech-to-Text, Azure Speech Services, and Amazon Transcribe offer high transcription accuracy, multilingual support, and robust streaming capabilities. However, they come with the following trade-offs:

- **Cost**: Pricing is based on transcription duration, which can be expensive for large-scale or real-time usage.
- **Data Privacy**: Audio is typically uploaded to external servers, raising privacy and regulatory concerns.
- **Limited Customization**: Proprietary APIs provide minimal access to model internals or training procedures.

In contrast, open-source models like Whisper and its variants (Faster-Whisper, WhisperX, Whisper.cpp) allow complete control over deployment, offline processing, and optimization for specific use cases.

## 6.3 Positioning of Selected Models

The evaluation identifies specific operational niches for each model based on the trade-offs observed during benchmarking:

- **Faster-Whisper (The Production Standard):** Despite lacking Metal acceleration on macOS, its industry-standard CTranslate2 backend makes it the strategic choice for **scalable cloud deployments** where reliability and integration are paramount.
- **WhisperX (The High-Precision Tool):** By integrating word-level alignment and diarization, it serves as the **premium solution** for offline academic archiving or research workflows where near-perfect transcription is the primary constraint.
- **Whisper.cpp (The Edge Solution):** Its ultra-low latency and minimal ”cold start” load time (**1.49s**) position it as the only viable candidate for strictly resource constrained edge devices or applications where speed is the overriding priority over literal accuracy.

## 7 Conclusion and Future Work

This project evaluated open-source ASR models on a MacBook Air M3 to identify a viable real-time transcription solution. Our findings confirm that while **Whisper.cpp** offers unparalleled speed, the accuracy degradation associated with its current quantization makes it unsuitable for complex academic content. Conversely, while **WhisperX** provides maximum precision, its resource footprint is heavy for a real-time streaming context.

**Limitations:** A key limitation of this study is its reliance on a single hardware architecture (Apple M3 Silicon). While this provides a representative benchmark for modern edge devices, performance metrics—specifically load times and processing speeds—will vary significantly on standard CPU architectures or systems with dedicated NVIDIA GPUs.

**Final Recommendation: Faster-Whisper** is selected as the optimal architecture for the final service. It provides the most robust balance of high-fidelity accuracy (**3.12% WER**) and a developer-friendly framework designed to scale efficiently on production-grade infrastructure.

**Future Work** will focus on:

- Integrating **speaker diarization** directly into the Faster-Whisper pipeline to achieve feature parity with WhisperX.
- Implementing **adaptive chunking** strategies to minimize end-to-end latency in continuous streaming scenarios.
- Benchmarking the service on **CUDA-enabled hardware** to empirically validate the projected performance gains in a server environment.
- Evaluation of Next-Generation Architectures: Incorporating the newly released **NVIDIA Nemotron-Speech-ASR (0.6B)** into the benchmarking framework. This model utilizes a cache-aware architecture that eliminates recomputation of audio windows, achieving a reported median latency of 24 ms.

  *However, this model was not prioritized for the current production phase due to two critical drawbacks: (1) it currently supports only English, which is insufficient for a multilingual academic environment; and (2) as a newly released architecture (January 2026), it lacks mature third-party tooling for integrated speaker diarization that is readily available within the Whisper ecosystem.*

The complete evaluation framework, along with the real-time transcription system built using **Faster-Whisper**, is publicly available on the project's GitLab repository: GitLab

## References

[1] A. Graves, A.-r. Mohamed, and G. Hinton, *Speech recognition with deep recurrent neural networks*, in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 6645–6649.

[2] G. Hinton et al., *Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups*, IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 82–97, 2012.

[3] Y. Zhang, T. Nguyen, and W.-N. Hsu, *Towards Real-Time and Low-Latency ASR for Live Captioning*, in *Proceedings of Interspeech*, 2023.

[4] L. Stapleton, A. Shaik, R. Xie, and N. Heffernan, *Real-Time Speech-to-Text for Accessibility: Evaluating Automatic Captioning in Classrooms*, in *Proceedings of the 21st International ACM SIGACCESS Conference on Computers and Accessibility*, 2020, pp. 408–410.

[5] SYSTRAN, *faster-whisper: Fast CPU/GPU inference of OpenAI Whisper models*, 2023, `https://github.com/SYSTRAN/faster-whisper`, accessed June 2025.

[6] M. Bain, *WhisperX: Enhanced word-level timestamps and speaker diarization*, 2023, `https://github.com/m-bain/whisperX`, accessed June 2025.

[7] G. Gerganov, *whisper.cpp: High-performance inference of Whisper models in C/C++*, 2023, `https://github.com/ggml-org/whisper.cpp`, accessed June 2025.

[8] OpenAI, *Whisper: Robust Speech Recognition via Large-Scale Weak Supervision*, 2022, `https://openai.com/research/whisper`, accessed August 2025.

[9] NVIDIA, *NVIDIA NeMo: A Scalable Generative AI Framework*, 2023, `https://developer.nvidia.com/nemo`, accessed August 2025.