Northeastern University

**CSYE7105 HIGH-PERFORMACE PARALLEL MACHINE LEARNING & AI**

**FINAL REPORT**

TEAM 19

PROJECT TITLE:

**High-Performance ASR System Development using Parallel GPU Computing**

MEMBERS:

**ARNAB CHAKRABORTY**

**RISHABH INDORIA**

# Introduction

The introduction of Automatic Speech Recognition (ASR) systems is revolutionizing human-computer interaction by enabling speech-driven applications such as virtual assistants and automated transcription services. The LibriSpeech dataset, consisting of audiobook recordings from a diverse set of speakers and featuring real-world ambient noises, serves as a comprehensive resource for training and validating ASR models. This project employs a hybrid CPU-GPU parallel processing approach using **joblib** and GPU-specific assignments to enhance the efficiency of model training, thereby accelerating development cycles.

## Background

Automatic Speech Recognition (ASR) has revolutionized communication by seamlessly transcribing spoken language into text. This innovation, coupled with the rise of Large Language Models (LLMs), has unlocked a multitude of applications:

* Summarization: ASR-generated transcripts enable the extraction of essential information from lengthy conversations or recordings, facilitating efficient summarization.
* Classification: Transcripts empower systems to classify conversations based on tasks and sentiment, aiding in contextual understanding and categorization.
* Question Answering: ASR transcripts provide input for question-answering systems, allowing users to pose questions in natural language and receive accurate responses.
* Pattern Mining: Rich ASR data enables pattern mining applications to identify user intents, recurring issues, and trends within conversations, offering valuable insights.
* Conversational AI (Artificial Intelligence): ASR technology underpins conversational AI systems, empowering virtual assistants and chatbots to understand and respond to user queries in real-time, enhancing user interactions.

The integration of ASR across various domains streamlines communication, improves user experience, and enhances decision-making processes.

## Motivation

Parallelization in Automatic Speech Recognition (ASR) is driven by the need for efficiency, scalability, and performance optimization.

Need for Parallelization:

* Faster Processing: Parallelization enables quicker processing of large audio files by distributing tasks across multiple processors or GPUs.
* Accelerated Training and Inferences: By leveraging parallelization techniques, ASR models can be trained faster and inference times reduced, leading to quicker development cycles and real-time applications.
* Memory Management: Parallelization helps overcome GPU memory limitations by distributing tasks across multiple GPUs, allowing for larger models and datasets without compromising performance.

In summary, parallelization in ASR enhances efficiency and scalability, driving advancements in technology and expanding its applications.

## Goals

Our primary objective is to enhance ASR system accuracy using state-of-the-art models and fine-tuning techniques on the LibriSpeech dataset. Specifically, we aim to compare two distinct models: Wav2Vec2, a prominent acoustic model by Facebook, and Whisper, an advanced model by OpenAI that incorporates language understanding capabilities. Through rigorous evaluation and fine-tuning, we seek to identify the model that achieves superior performance in transcription accuracy and semantic understanding.

# Methodology

**Procuring Compute Instance:**

We recognize the necessity for a high-processing compute instance and opt for the ml.g4dn.12xlarge instance provided by AWS (Amazon Web Services). This instance, designed for machine learning tasks demanding efficient GPU acceleration, boasts impressive specifications:

* **Instance Family**: G4 instances
* **GPU**: NVIDIA T4 Tensor Core GPU
* **vCPU**: 48
* **Memory** (GiB): 192
* **Storage**: EBS only
* **Network Performance**: Up to 25 Gbps
* **Processor Architecture**: 64-bit
* **Operating System**: Linux

A computer screen shot of a computer

Description automatically generatedImage Display GPUs in Action

**Exploratory Data Analysis (EDA):**

We conduct extensive EDA, evaluating the diversity of speakers, waveform lengths, and other pertinent factors. Our observations reveal multiple speakers, each with a minimum of 35 samples. Also, most of our data spans less than 5 seconds, prompting us to process them in batches.

**Data Pre-processing:**

We use Joblib's parallel processing, to truncate and pad the audio files to 6 Seconds (waveform = 96,000) in the LibriSpeech dataset to enable batch processing of audio files.

# Model Evaluation:

**Data Loader**:

We leverage parallel processing by employing a data Loader to iterate over our dataset, thereby accelerating the training process through workload distribution across multiple cores or GPUs.

**CPU (Central Processing Unit) Parallelization:**

**Employ Whisper-tiny model multiple CPUs (Central Processing Unit) using Joblib**

* Model: Whisper Tiny
* By: OpenAI
* No. of parameters: 35M
* Batch size = 32

Table Displaying Speed Up and Efficiency achieved using Multiprocessing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No. of CPUs | **1** | **4** | **8** | **16** | **32** | **48** |
| Time (s) | 310 | 172 | 144 | 142 | 149 | 149 |
| Speed Up | 1 | 1.8 | 2.15 | 2.17 | 2.07 | 2.08 |
| Efficiency | 1 | 0.45 | 0.26 | 0.14 | 0.06 | 0.04 |

**Speedup Plot**

A graph with blue dots and a line

Description automatically generated

**Efficiency Plot**

A graph with a line and a dotted line

Description automatically generated

**Evaluating GPU Parallelization**

Parallelization of Models: Maximizing computational resources, we parallelize the Wav2Vec2 and Whisper models, enabling simultaneous transcript generation on both CPU and GPU. This parallelization is achieved through the Joblib ensuring efficient utilization of available hardware resources.

**Model 1: Whisper Base by OpenAI**

* No. of parameters: 70M

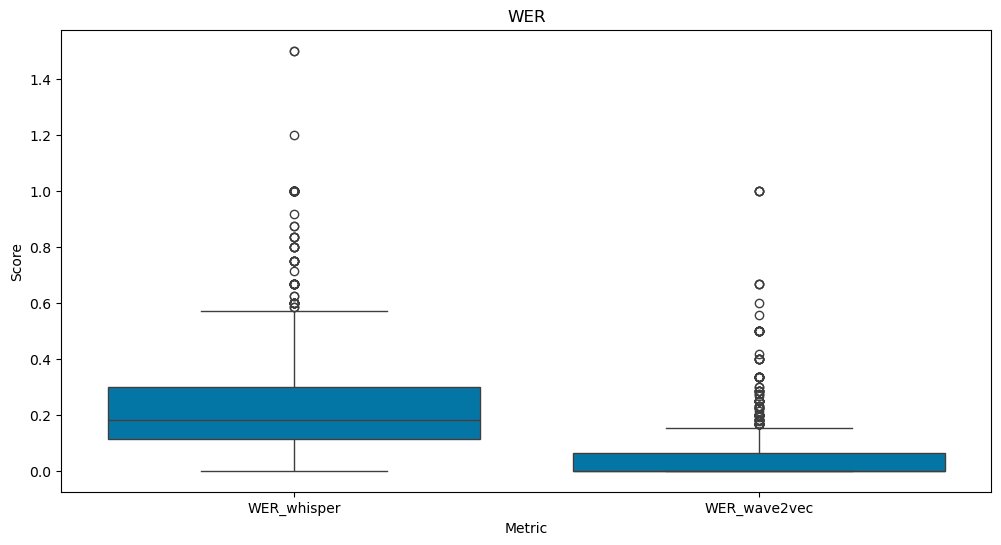
**Model 2: Wav2Vec2 by Facebook**

* No. Of parameters: 90M
* Number of GPUs: 4

Table showing Speedup and Efficiency achieved using multi-processing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Computation Time (s)** | **1 GPU** | **4 GPU** | **Speedup** | **Efficiency** |
| Whisper | 249.12 | 70.53 | 3.5x | 88% |
| Wave2Vec2 | 43.90 | 18.05 | 2.4x | 60% |

WER Rate Calculation: We compute the mean Word Error Rate (WER) to assess model performance, facilitating insightful evaluation of transcription accuracy.



Transcript Evaluation

**Original Transcription**

* MISTER QUILTER IS THE APOSTLE OF THE MIDDLE CLASSES AND WE ARE GLAD TO WELCOME HIS GOSPEL
* NOR IS MISTER QUILTER'S MANNER LESS INTERESTING THAN HIS MATTER
* ON THE GENERAL PRINCIPLES OF ART MISTER QUILTER
* WRITES WITH EQUAL LUCIDITY

Whisper Transcriptions

* Mr. Quilter is the apostle of the middle classes, and we are glad to welcome his gospel.
* Nor is Mr. Quilter's manner less interesting than his matter.
* On the general principles of art
* Mr. Krilter writes with equal lucidity

**Wav2Vec2 Transcriptions**

* MISTER QUILTER IS THE APOSTLE OF THE MIDDLE CLASSES AND WE ARE GLAD TO WELCOME HIS GOSPEL
* NOR IS MISTER QUILTER'S MANNER LESS INTERESTING THAN HIS MATTER
* ON THE GENERAL PRINCIPLES OF ART MISTER QUILTER
* WRITES WITH EQUAL LUCIDITY

Observation:

* The Whisper model outperforms Wav2Vec2 in capturing transcript semantics despite a higher word error rate.
* Wav2Vec2 has been trained on LibriSpeech data, while Whisper has not been exposed to it, highlighting Whisper's adaptability.
* Normalizing transcripts can notably reduce the Word Error Rate by treating similar variations consistently, like "Mr." and "Mister."

Model Fine-Tuning:

We Fine-tune the Whisper tiny model as it can be easily used on Edge Devices and would have a significantly quicker inference time. We utilize Torch Distributed Data Parallel to leverage GPU computing in Parallel which significantly expedites the training process.

# Results

We Observe That Fine-tuned Whisper tiny model significantly outperforms on the dev clean dataset with quicker inference time.

A graph with a line

Description automatically generated

Avg WER vs epochs

A graph with a line

Description automatically generated

Loss Curve vs epochs

|  |  |  |
| --- | --- | --- |
| **Model** | **Time(s)** | **Avg WER** |
| Tiny | 43 | 0.34 |
| Base | 59 | 0.33 |
| Large | 140 | 0.32 |
| Fine-tuned Tiny | 43 | 0.16 |

# Conclusion

The project demonstrated the efficacy of parallel GPU computing in Fine-Tuning a small model to drastically outperform large models and captures semantics.

**Next Steps**

* Perform Normalization before WER to capture results effectively
* Fine-tune Whisper on Noisy datasets like LibriSpeech, partition dev-other
* Compare against other SOTA ASR models like Canary, Parakeet & Jasper offered by Nvidia

# References

[1] LibriSpeech dataset: <https://www.openslr.org/12>[2] Hugging Face Datasets: <https://huggingface.co/datasets/librispeech_asr>[3] PyTorch: [https://pytorch.org](https://pytorch.org/)

[4] Whisper: <https://arxiv.org/pdf/2212.04356.pdf>

[5] Wav2vec2: <https://arxiv.org/pdf/2006.11477.pdf>

[6] Joblib: <https://joblib.readthedocs.io/en/stable/>