



**HCMUS**  
Viet Nam National University  
Ho Chi Minh City  
University of Science

**Khoa Công nghệ Thông tin**

**Nhập môn học máy**

---

# MODEL REPORT

---

**Ngày 15 tháng 12 năm 2025**

**Đồ án được thực hiện bởi**

Đặng Anh Kiệt - 23127077 - 23KHMT

Phạm Minh Triết - 23127132 - 23KHMT

Trần Quang Phúc - 23127302 - 23KHMT

Kiều Duy Hiếu - 23127365 - 23KHMT

**Supervised by**

Bùi Tiến Lên

Lê Nhật Nam

Võ Nhật Tân

# Mục lục

<b>1</b>	<b>Giới thiệu bài toán</b>	<b>4</b>
<b>2</b>	<b>Tổng quan dữ liệu đầu vào</b>	<b>5</b>
2.1	Giới thiệu bộ dữ liệu VIVOS . . . . .	5
2.1.1	Tỷ lệ chia dữ liệu . . . . .	6
2.1.2	Tiền xử lý . . . . .	6
<b>3</b>	<b>Lựa chọn Mô hình và Kiến trúc</b>	<b>8</b>
3.1	Mô hình sử dụng . . . . .	8
3.2	Lí do lựa chọn . . . . .	8
3.2.1	PhoWhisper . . . . .	9
3.2.2	Wav2Vec2 . . . . .	9
3.3	Kiến trúc chi tiết . . . . .	10
3.3.1	OpenAI Whisper . . . . .	10
3.3.2	PhoWhisper . . . . .	11
3.3.3	Wav2Vec2 . . . . .	12
<b>4</b>	<b>Cấu hình huấn luyện</b>	<b>13</b>
4.1	Wav2Vec2 . . . . .	13
4.1.1	Hàm mất mát . . . . .	13
4.1.2	Thuật toán tối ưu . . . . .	14
4.1.3	Siêu tham số . . . . .	14
4.2	PhoWhisper . . . . .	14
4.2.1	Hàm mất mát . . . . .	14
4.2.2	Thuật toán tối ưu . . . . .	14
4.2.3	Siêu tham số . . . . .	15

4.3	OpenAI Whisper . . . . .	15
4.3.1	Hàm mất mát . . . . .	15
4.3.2	Thuật toán tối ưu . . . . .	15
4.3.3	Siêu tham số . . . . .	16

# Chương 1

## Giới thiệu bài toán

Trong lĩnh vực Xử lý Ngôn ngữ Tự nhiên (NLP) và Xử lý Tín hiệu, dự án này giải quyết bài toán **Nhận dạng Tiếng nói Tiếng Việt Tự động (Vietnamese Automatic Speech Recognition - ASR)**. Mục tiêu chính là xây dựng một hệ thống học máy có khả năng chuyển đổi tín hiệu giọng nói của con người thành dạng văn bản tương ứng với độ chính xác cao.

Một cách hình thức, bài toán ASR có thể được định nghĩa là tìm chuỗi từ có khả năng xảy ra cao nhất  $\hat{W}$  với đầu vào là chuỗi các đặc trưng âm thanh  $X$ . Bài toán này được mô hình hóa dưới dạng bài toán phân loại xác suất, trong đó chúng ta tìm cách tối đa hóa xác suất hậu nghiệm:

$$\hat{W} = \underset{W}{\operatorname{argmax}} P(W|X) \quad (1.1)$$

Trong đó:

- $X = \{x_1, x_2, \dots, x_T\}$  đại diện cho chuỗi các vector đặc trưng âm thanh được trích xuất từ đầu vào âm thanh thô (ví dụ: MFCC hoặc Spectrogram).
- $W = \{w_1, w_2, \dots, w_N\}$  đại diện cho chuỗi các từ trong câu mục tiêu.
- $P(W|X)$  là xác suất của chuỗi từ  $W$  khi biết quan sát âm thanh  $X$ .

## Chương 2

# Tổng quan dữ liệu đầu vào

### 2.1 Giới thiệu bộ dữ liệu VIVOS

- **Tổng số giờ audio:** 15 giờ
- **Tần số lấy mẫu:** 16,000 Hz (16 kHz)
- **Định dạng file:** WAV
- **Ngôn ngữ:** Tiếng Việt (giọng miền Nam - Southern Vietnamese)
- **Môi trường ghi âm:** Phòng yên tĩnh với microphone chất lượng cao
- **Nội dung:** Người đọc đọc từng dòng văn bản đã chuẩn bị sẵn
- Mô hình `nguyenvulebinh/wav2vec2-base-vietnamese-250h` sau đó được fine-tune trên VIVOS trong nghiên cứu này.
- Mô hình `vinai/PhoWhisper-small` được VinAI pretrain trên lượng dữ liệu tiếng Việt lớn, sau đó được fine-tune trên VIVOS.
- Mô hình `openai/whisper-tiny` được OpenAI pretrain trên lượng dữ liệu đa ngôn ngữ lớn, sau đó được fine-tune trên VIVOS.
- Việc sử dụng pretrained models giúp tận dụng kiến thức đã học từ dữ liệu lớn, chỉ cần fine-tune trên VIVOS 15 giờ để đạt hiệu quả cao.

### 2.1.1 Tỷ lệ chia dữ liệu

Bộ dữ liệu VIVOS đã được chia sẵn thành hai tập: `train` và `test`. Tỷ lệ chia cụ thể như sau:

Bảng 2.1: Phân chia dữ liệu VIVOS

Tập dữ liệu	Số lượng mẫu	Tỷ lệ (%)
Training (Train)	11,660	93.88%
Testing (Test)	760	6.12%
<b>Tổng cộng</b>	<b>12,420</b>	<b>100%</b>

**Lưu ý về Validation Set:** Trong quá trình fine-tuning PhoWhisper, 200 mẫu từ tập Test được sử dụng làm validation để theo dõi quá trình huấn luyện.

**Lý do chọn tỷ lệ này:**

1. **Sử dụng tỷ lệ có sẵn của VIVOS:** Bộ dữ liệu VIVOS đã được các nhà phát triển chia sẵn theo tỷ lệ  $\approx 94/6$  (train/test). Việc giữ nguyên cách chia này đảm bảo tính nhất quán với các nghiên cứu trước đó sử dụng cùng bộ dữ liệu.
2. **Tối đa hóa dữ liệu huấn luyện:** Với tổng số 12,420 mẫu, việc dành phần lớn dữ liệu (93.88%) cho huấn luyện giúp mô hình học được nhiều biến thể ngôn ngữ và giọng nói hơn.
3. **Tập Test đủ lớn để đánh giá:** 760 mẫu test đủ để đánh giá hiệu năng mô hình một cách đáng tin cậy, bao gồm nhiều speaker và nội dung khác nhau.

### 2.1.2 Tiền xử lý

**Xử lý Audio:**

1. **Tải file âm thanh:** Đọc file WAV từ thư mục `waves/` theo cấu trúc `waves/SPEAKER_ID/FILE_ID.wav`.
2. **Resampling:** Chuyển đổi tần số lấy mẫu về 16,000 Hz (16 kHz) - là tần số của vivos
3. **Feature Extraction:**
  - **PhoWhisper và Whisper-tiny:** Chuyển đổi audio thành Log-Mel Spectrogram bằng `WhisperProcessor.feature_extractor()`. Spectrogram này là biểu diễn 2D của tín hiệu âm thanh theo thời gian và tần số.

- **Wav2Vec2:** Sử dụng `Wav2Vec2FeatureExtractor` để chuẩn hóa waveform thành `input_values` với padding phù hợp.

### Xử lý Text (Nhãn):

1. **Đọc transcript:** Tải nội dung văn bản từ file `prompts.txt`, mỗi dòng có định dạng:  
`FILE_ID câu_văn_bản.`

2. **Normalization:** Chuẩn hóa văn bản về chữ thường (lowercase) tiếng Việt:

```
1 ref_norm = text.lower().strip()
2 pred_norm = transcription.lower().strip()
3
```

3. **Tokenization:**

- **PhoWhisper và Whisper-tiny:** Sử dụng `WhisperProcessor.tokenizer()` để chuyển văn bản thành chuỗi token IDs, lưu vào trường `labels`.
- **Wav2Vec2:** Sử dụng `Wav2Vec2CTCTokenizer` với các token đặc biệt: `[UNK]` (unknown), `[PAD]` (padding), và `|` (word delimiter).

**Data Collation:** Trong quá trình huấn luyện, các batch dữ liệu được xử lý bởi Data Collator:

- **Padding:** Các input features và labels được pad về cùng độ dài trong mỗi batch.
- **Masking:** Padding tokens trong labels được thay bằng giá trị `-100` để không tính vào hàm loss:

```
1 labels = labels.masked_fill(attention_mask.ne(1), -100)
2
```

- **Loại bỏ BOS token:** Token bắt đầu câu (BOS - Beginning of Sentence) được loại bỏ khỏi labels nếu có.

# Chương 3

## Lựa chọn Mô hình và Kiến trúc

### 3.1 Mô hình sử dụng

- OpenAI Whisper
- PhoWhisper
- Wav2Vec2

### 3.2 Lí do lựa chọn

#### OpenAI Whisper

Đại diện cho SOTA (State-of-the-art) và Tính Đa Nhiệm. Đây là mô hình "Baseline" (cơ sở) mạnh mẽ nhất hiện nay để so sánh.

- Kiến trúc Transformer quy mô lớn: Whisper sử dụng kiến trúc Encoder-Decoder (Seq2Seq) được huấn luyện trên 680.000 giờ dữ liệu đa ngôn ngữ. Chọn Whisper cho phép bạn tiếp cận công nghệ tiên tiến nhất hiện nay.
- Khả năng Robustness (Chống nhiễu): Không giống các mô hình cũ, Whisper cực kỳ mạnh mẽ trong việc xử lý âm thanh nhiễu, giọng địa phương và ngữ điệu tự nhiên mà không cần tiền xử lý quá phức tạp.
- Zero-shot Learning: Chọn Whisper để kiểm chứng khả năng nhận dạng tiếng Việt của một mô hình đa ngôn ngữ (Multilingual) mà không cần fine-tune lại xem hiệu suất gốc của nó tốt đến mức nào.



### 3.2.1 PhoWhisper

Đại diện cho sự "Thích nghi ngôn ngữ" (Language Adaptation). Đây là phiên bản cải tiến chuyên biệt, chứng minh tầm quan trọng của việc tối ưu hóa cho tiếng Việt.

- Fine-tuning cho tiếng Việt: PhoWhisper là phiên bản Whisper được fine-tune trên tập dữ liệu tiếng Việt lớn. Lý do chọn nó là để chứng minh giả thuyết: "Một mô hình lớn đa ngôn ngữ (Whisper) sẽ hoạt động tốt hơn nếu được tinh chỉnh sâu trên dữ liệu bản địa."
- Khắc phục nhược điểm của Whisper gốc: Whisper gốc đôi khi gặp lỗi về dấu câu hoặc từ vựng đặc thù của Việt Nam. PhoWhisper giải quyết vấn đề này. Chọn mô hình này giúp đồ án của bạn có tính ứng dụng thực tế cao hơn tại Việt Nam.

### 3.2.2 Wav2Vec2

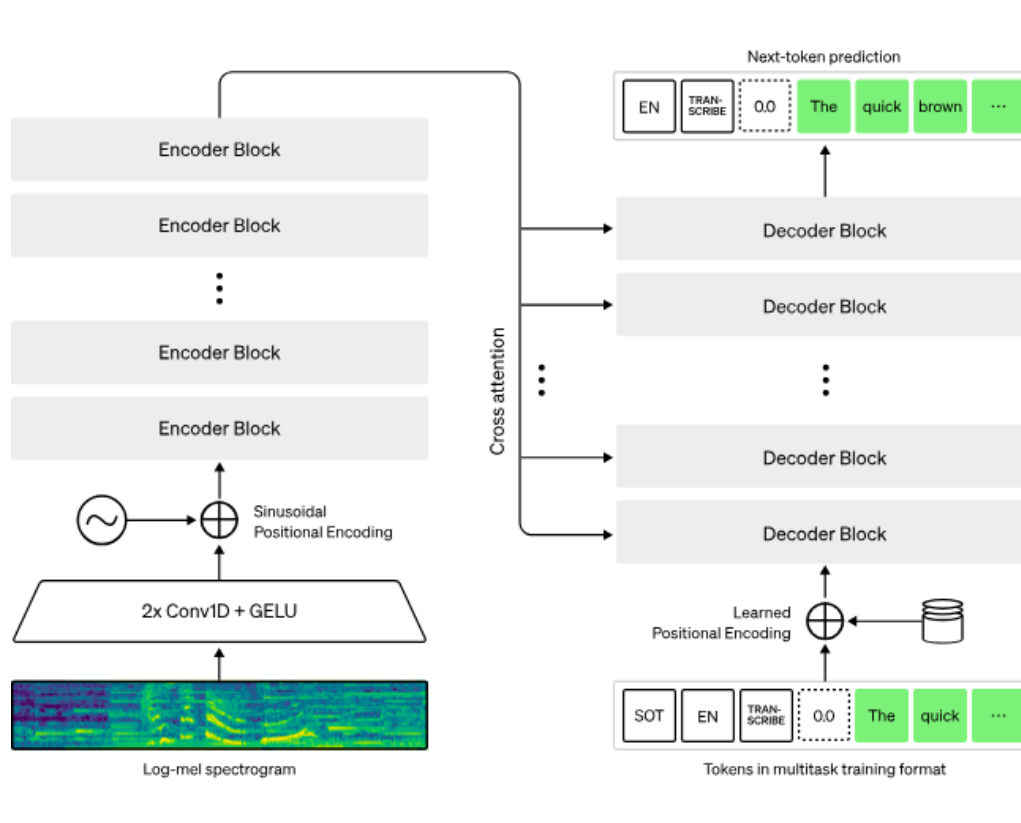
Đại diện cho Kiến trúc Self-Supervised Learning (SSL). Đây là đối trọng về mặt kiến trúc và hiệu suất tính toán.

- Sự khác biệt về kiến trúc (Encoder-only vs Encoder-Decoder): Trong khi Whisper là Encoder-Decoder, Wav2Vec2 sử dụng cơ chế CTC (Connectionist Temporal Classification) và Self-Supervised Learning. Việc chọn Wav2Vec2 giúp đồ án có sự so sánh đa dạng về mặt kỹ thuật chứ không chỉ so sánh các biến thể của cùng một mô hình.
- Tối ưu tài nguyên và tốc độ: Wav2Vec2 thường nhẹ hơn và có độ trễ (latency) thấp hơn Whisper trong suy luận (inference). Đây là lựa chọn lý tưởng để so sánh về khía cạnh hiệu năng triển khai (Deployment efficiency) trên các thiết bị giới hạn tài nguyên.
- Tiền thân của công nghệ ASR hiện đại: Trước khi Whisper ra mắt, Wav2Vec2 là chuẩn mực (SOTA). Việc đưa nó vào giúp bạn so sánh được sự tiến hóa của công nghệ.

## 3.3 Kiến trúc chi tiết

### 3.3.1 OpenAI Whisper

#### Sơ đồ kiến trúc



Hình 3.1: Sơ đồ kiến trúc OpenAI Whisper

#### Số lượng tham số

Sử dụng mô hình kích thước **tiny** với **39 Triệu** tham số.

#### Hàm kích hoạt

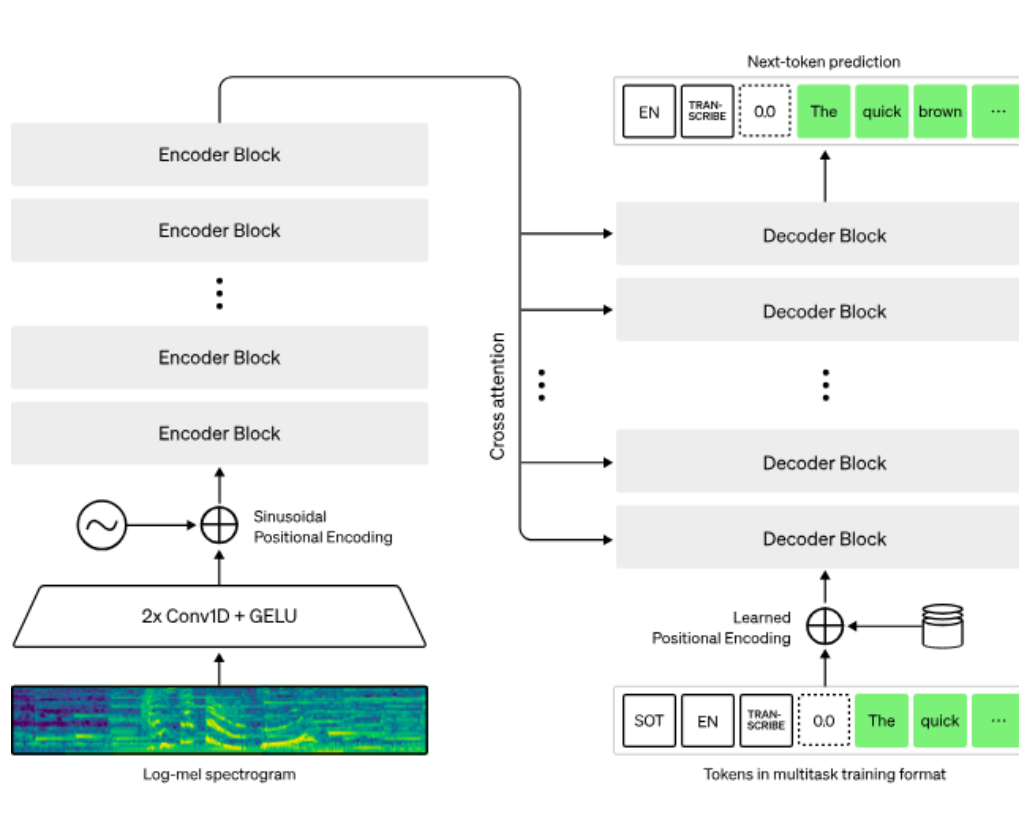
Bao gồm 2 phần:

- **2 x Conv1d và GELU activation function:** dùng để trích xuất các đặc trưng từ log-Mel spectrogram đầu vào.
- **Positional Embedding:** Whisper sử dụng sinusoidal positional embedding cho phép mã hóa vị trí và vị trí của từng token và vị trí tương đối của từng token với nhau.

### 3.3.2 PhoWhisper

Vì PhoWhisper là bản fine-tune của OpenAI Whisper nên kiến trúc, số lượng tham số và hàm kích hoạt đều tương tự OpenAI Whisper.

#### Sơ đồ kiến trúc



Hình 3.2: Sơ đồ kiến trúc PhoWhisper

#### Số lượng tham số

Sử dụng mô hình kích thước **tiny** với **39 Triệu** tham số.

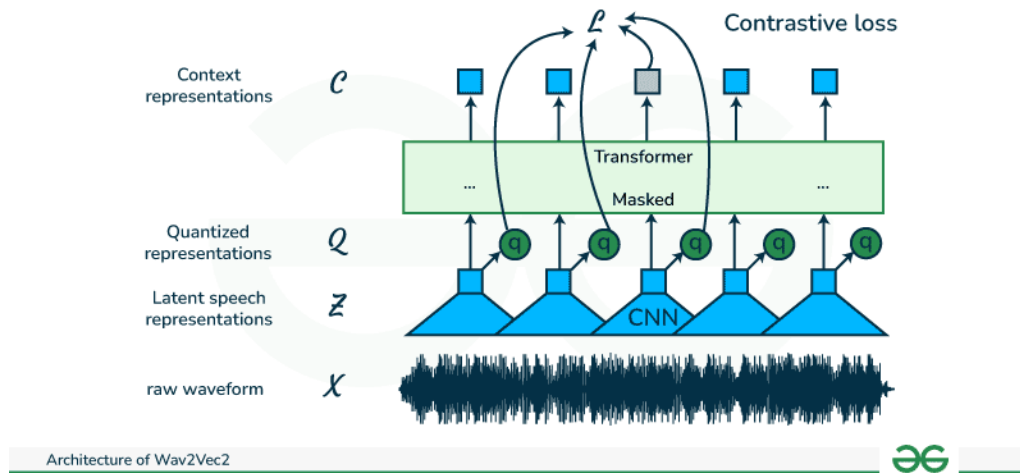
#### Hàm kích hoạt

Bao gồm 2 phần:

- **2 x Conv1d và GELU activation function:** dùng để trích xuất các đặc trưng từ log-Mel spectrogram đầu vào.
- **Positional Embedding:** Whisper sử dụng sinusoidal positional embedding cho phép mã hóa vị trí và vị trí của từng token và vị trí tương đối của từng token với nhau.

### 3.3.3 Wav2Vec2

#### Sơ đồ kiến trúc



Hình 3.3: Sơ đồ kiến trúc Wav2Vec2

#### Số lượng tham số

Sử dụng mô hình kích thước **base** với **95 Triệu** tham số.

#### Hàm kích hoạt

Bao gồm 2 phần:

- **2 x Conv1d và GELU activation function:** dùng để trích xuất các đặc trưng từ log-Mel spectrogram đầu vào.
- **Positional Embedding:** Whisper sử dụng sinusoidal positional embedding cho phép mã hóa vị trí và vị trí của từng token và vị trí tương đối của từng token với nhau.

# Chương 4

## Cấu hình huấn luyện

### 4.1 Wav2Vec2

#### 4.1.1 Hàm mất mát

Sử dụng phối hợp giữa **InfoNCE Loss**, **Diversity Loss** và **CTC Loss** nhằm giải quyết nút thắt lớn nhất của Deep Learning là thiếu dữ liệu dán nhãn.

##### **InfoNCE Loss**

Được dùng trong giai đoạn **Self-Supervised Pretraining**.

**Tại sao dùng?** InfoNCE buộc mô hình phải hiểu nội dung cao cấp (như âm vị, ngữ điệu) để phân biệt đâu là đoạn âm thanh "đúng" và đâu là nhiễu, thay vì chỉ học vẹt các chi tiết ở mức thấp (low-level signal).

##### **Diversity Loss**

Được dùng trong giai đoạn **Self-Supervised Pretraining**.

**Tại sao dùng?** Để ngăn chặn hiện tượng "Codebook Collapse" (hoặc Mode Collapse).

##### **CTC Loss (Connectionist Temporal Classification)**

Được dùng trong giai đoạn **Supervised Fine-tuning**.

**Tại sao dùng?** Đây là tiêu chuẩn vàng cho các bài toán Sequence-to-Sequence trong ASR khi không có thông tin căn chỉnh (alignment info) từ trước.

### 4.1.2 Thuật toán tối ưu

**Optimizer:** AdamW (mặc định HF Trainer).

**Learning rate / Scheduler:** Có scheduler (warm-up lên rồi decay)

### 4.1.3 Siêu tham số

**Siêu tham số:**

- train\_batch\_size: 4
- num\_train\_epochs: 5
- max\_steps: 3645
- logging\_steps: 50
- save\_steps: 400
- eval\_steps: 200
- global\_step: 3645

**Phương pháp tinh chỉnh siêu tham số:** thủ công

## 4.2 PhoWhisper

### 4.2.1 Hàm mất mát

Sử dụng **Cross-Entropy Loss** trên đầu ra sequence-to-sequence của Whisper; padding token được thay bằng (-100) để không tính vào loss (xem phowhisper.py). Đánh giá bằng **WER**.

### 4.2.2 Thuật toán tối ưu

**Optimizer:** AdamW (mặc định HF Trainer).

**Learning rate / Scheduler:** có sử dụng scheduler decay.

### 4.2.3 Siêu tham số

**Siêu tham số:**

- per\_device\_train\_batch\_size: 4
- gradient\_accumulation\_steps: 4
- learning\_rate: 1e-3
- max\_steps: 500
- fp16: True
- save\_steps: 100, eval\_steps: 100, logging\_steps: 50
- dataloader\_num\_workers: 0

**Tinh chỉnh (LoRA):**

- r=32
- lora\_alpha=64
- target\_modules=["q\_proj", "v\_proj"]
- lora\_dropout=0.05
- bias="none"

**Phương pháp tinh chỉnh siêu tham số:** thủ công

## 4.3 OpenAI Whisper

### 4.3.1 Hàm mất mát

Sử dụng **Cross-Entropy Loss** trên đầu ra sequence-to-sequence của Whisper.

**Đánh giá:** WER.

### 4.3.2 Thuật toán tối ưu

**Optimizer:** AdamW (mặc định HF Trainer).

**Learning rate / Scheduler:** sử dụng warm-up lên rồi decay.

### 4.3.3 Siêu tham số

**Siêu tham số:**

- per\_device\_train\_batch\_size: 16
- gradient\_accumulation\_steps: 1
- learning\_rate: 1e-5
- warmup\_steps: 100
- num\_train\_epochs: 15
- eval\_strategy: epoch
- save\_strategy: epoch
- save\_total\_limit: 2
- max\_steps: -1 (disabled)
- logging\_steps: 50
- fp16: True
- per\_device\_eval\_batch\_size: 8
- generation\_max\_length: 225



- report\_to: tensorboard
- load\_best\_model\_at\_end: True
- metric\_for\_best\_model: wer
- greater\_is\_better: False
- dataloader\_num\_workers: 2
- dataloader\_pin\_memory: True
- remove\_unused\_columns: False

**Tính chỉ số tham số:** thủ công.

# Tài liệu tham khảo

- [1] Andrew Bawitlung, Sandeep Kumar Dash, and Radha Mohan Pattanayak. *Mizo Automatic Speech Recognition: Leveraging Wav2vec 2.0 and XLS-R for Enhanced Accuracy in Low-Resource Language Processing*. URL: <https://dl.acm.org/doi/full/10.1145/3746063>.
- [2] GeeksforGeeks. *Wav2Vec2: Self-A Supervised Learning Technique for Speech Representations*. URL: <https://www.geeksforgeeks.org/nlp/wav2vec2-self-a-supervised-learning-technique-for-speech-representations/>.
- [3] Nguyễn Văn Mạnh. *[Voice-Speech to text] Khám phá Whisper và công cụ API chuyển đổi âm thanh thành văn bản Deepgram*. URL: <https://viblo.asia/p/voice-speech-to-text-kham-pha-whisper-va-cong-cu-api-chuyen-doi-am-thanh-thanh-van-ban-deepgram-AZoJjd0yVY7>.
- [4] OpenAI. *Introducing Whisper*. URL: <https://openai.com/index/whisper/>.
- [5] OpenAI. *OpenAI Whisper Repo*. URL: <https://github.com/openai/whisper/blob/main/README.md>.