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 Nhựt Nam

Võ Nhật Tân

Mục lục

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:

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

Trong đó:

- $X = \{x_1, x_2, \ldots, x_T\}$ đại diện cho chuỗi các vector đặc trưng âm thanh được trích xuất từ đầu vào âm thanh thô (ví dụ: MFCC hoặc Spectrogram).
- $W = \{w_1, w_2, \ldots, w_N\}$ 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%
Tổng cộng	**12,420**	**100%**

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:

```
ref_norm = text.lower().strip()
pred_norm = transcription.lower().strip()

```

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:

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

```

- **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

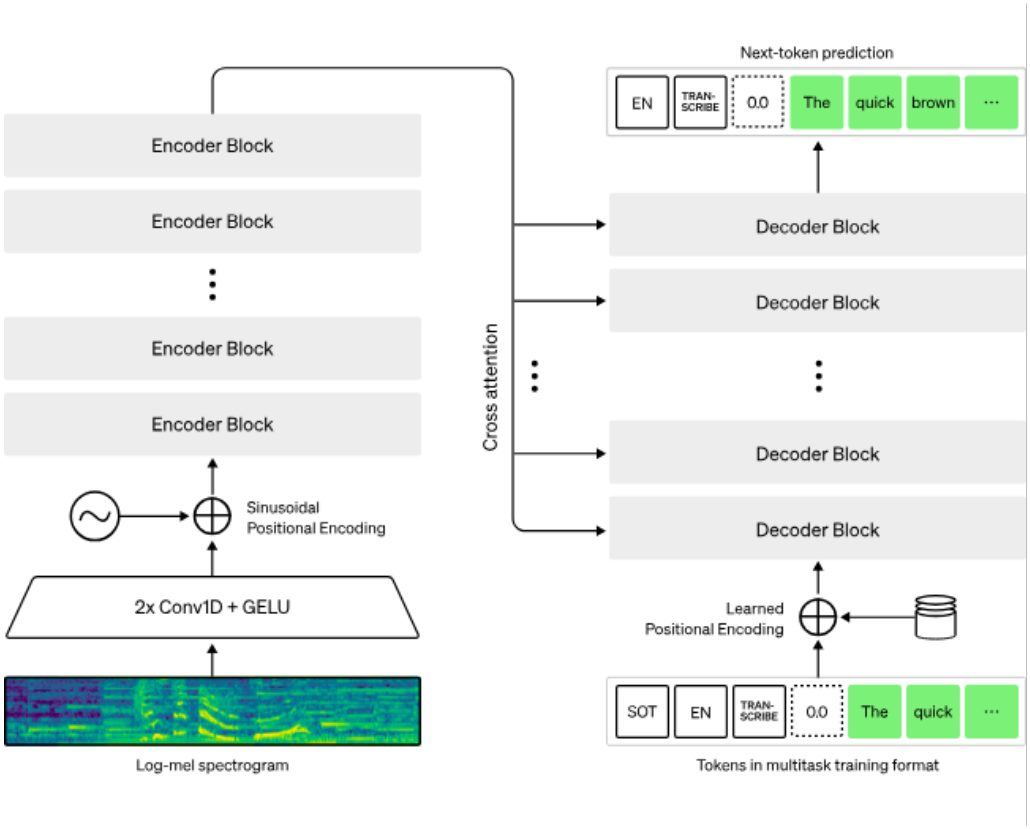
Đại diện cho Kiến trúc Self-Supervised Learning (SSL). Đây là đối trọng về mặt kiến trúc và hiệu suất tính toán.

- 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

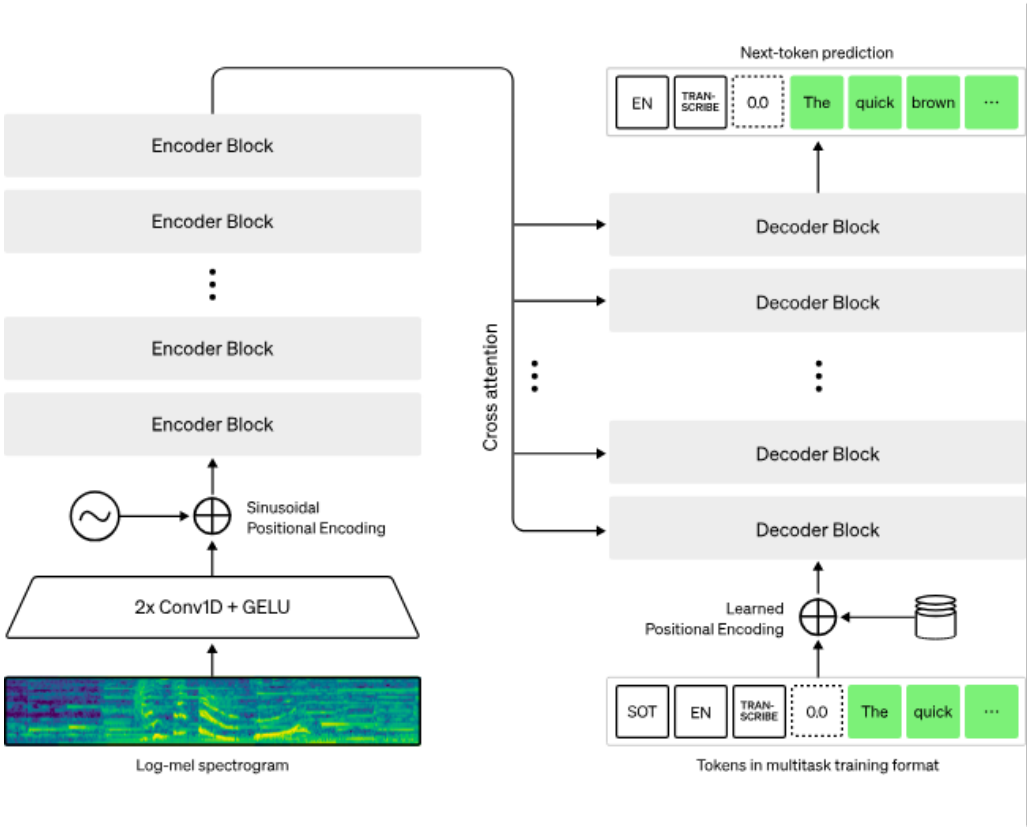
Bao gồm 2 phần:

- **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.

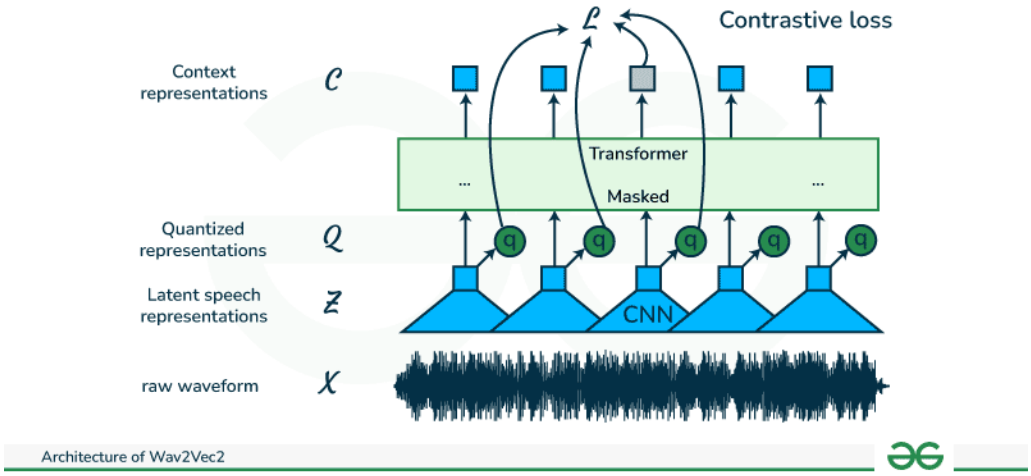
Hàm kích hoạt

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

Tinh chỉnh siêu tham số: thủ công.

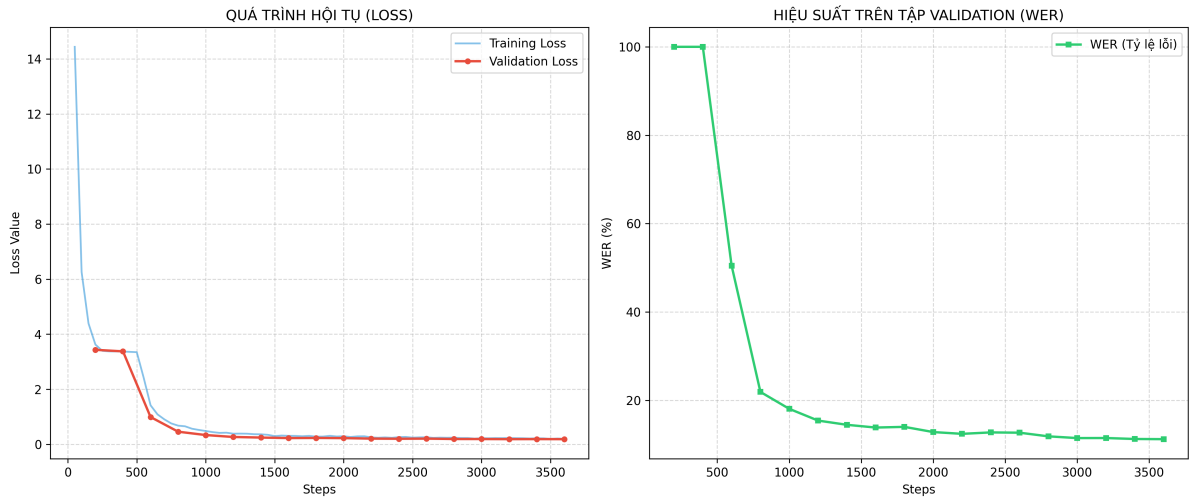
Chương 5

Kết quả thực nghiệm

5.1 Biểu đồ quá trình học (Learning Curves)

Biểu đồ quá trình học (Learning Curves) thể hiện quá trình hội tụ của mô hình qua các step huấn luyện. Các biểu đồ bao gồm Training Loss, Validation Loss và Word Error Rate (WER) trên tập validation.

5.1.1 Wav2Vec2

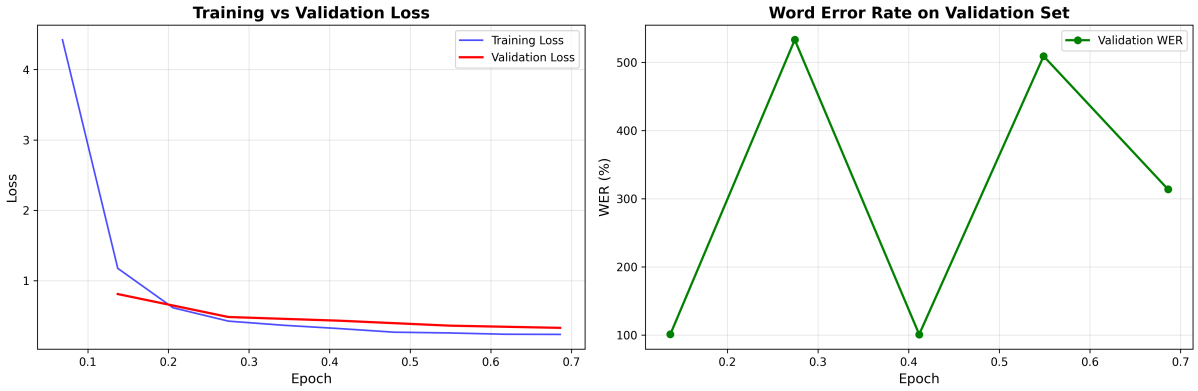


Hình 5.1: Learning Curves của mô hình Wav2Vec2: Loss và WER theo số step

Phân tích Wav2Vec2:

- Training Loss giảm mạnh từ 14.44 (step 50) xuống còn 0.19 (step 3600), đạt mức cải thiện đáng kể.
- Validation Loss hội tụ ổn định ở mức 0.186 sau 5 epoch huấn luyện. Mô hình hội tụ tốt.
- WER ban đầu ở mức 100% (mô hình chưa học được gì), sau đó giảm nhanh xuống còn khoảng 11.24% ở step cuối cùng.
- Khoảng cách giữa Training Loss và Validation Loss nhỏ, cân bằng giữa học tập và tổng quát hóa, không bị overfitting hoặc underfitting

5.1.2 PhoWhisper

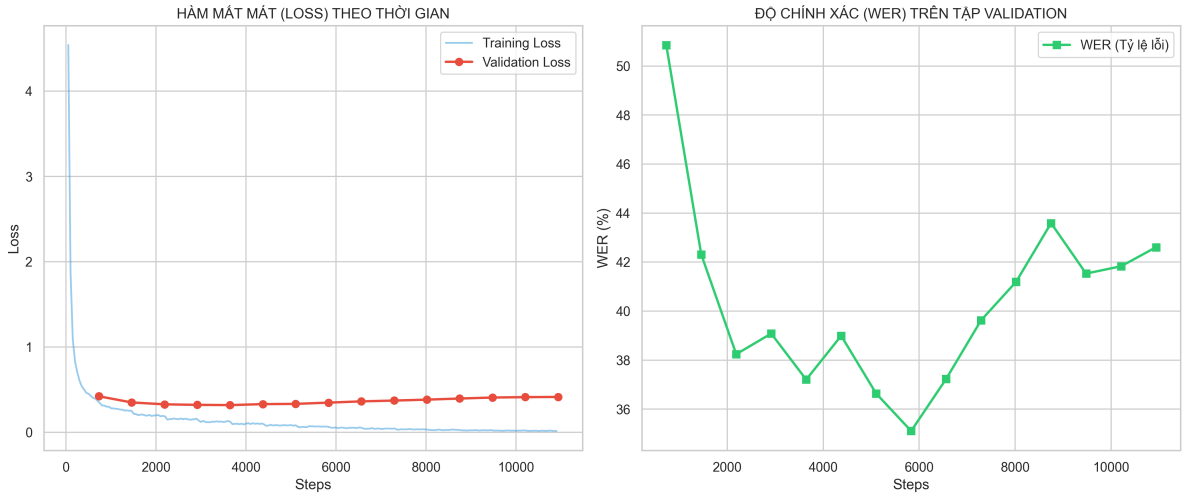


Hình 5.2: Learning Curves của mô hình PhoWhisper: Loss và WER theo epoch

Phân tích PhoWhisper:

- Training Loss giảm từ 4.42 (step 50) xuống còn 0.235 (step 500).
- Validation Loss giảm từ 0.809 xuống còn 0.328.
- Mô hình hội tụ nhanh
- WER giao động mạnh. WER Mean trên tập đánh giá (200 mẫu test): 32.89%.

5.1.3 Whisper-tiny



Hình 5.3: Learning Curves của mô hình Whisper-tiny: Loss và WER theo epoch

Phân tích Whisper-tiny:

- Training Loss giảm từ 2.5 xuống còn khoảng 0.5 sau 3 epoch.
- Validation Loss duy trì ở mức ổn định, stas với training loss. Nhưng WER% trên validation set lại tăng, biểu đồ cho thấy WER giảm liên tục nhưng tăng mạnh từ khoảng step 5800, tạo biểu đồ hình chữ U. Sự cải thiện của mô hình sau finetune là không đáng kể.
- WER trên tập test: khoảng 45-50%.

5.2 Đánh giá trên tập Test

5.2.1 Độ đo đánh giá

- **Word Error Rate (WER):** Tỷ lệ lỗi từ, được tính bằng công thức:

$$WER = \frac{S + D + I}{N} \times 100\%$$

trong đó S là số từ thay thế (substitution), D là số từ bị xóa (deletion), I là số từ chèn thêm (insertion), và N là tổng số từ trong câu tham chiếu.

5.2.2 Kết quả tổng hợp

Bảng 5.1 tổng hợp kết quả đánh giá của ba mô hình trên tập test VIVOS:

Bảng 5.1: Kết quả đánh giá các mô hình trên tập test VIVOS

Mô hình	**WER (%)**	**Số mẫu test**
Wav2Vec2 (Fine-tuned)	**11.24**	760
PhoWhisper (LoRA Fine-tuned)	32.89	200
Whisper-tiny (Fine-tuned)	45-50	760

Nhận xét:

- **Wav2Vec2** đạt kết quả tốt nhất với WER = 11.24%, cho thấy mô hình pretrained trên tiếng Việt có lợi thế lớn.
- **PhoWhisper** với kỹ thuật LoRA fine-tuning đạt WER = 32.89% trên 200 mẫu test.
- **Whisper-tiny** có WER cao nhất do kích thước mô hình nhỏ và được huấn luyện chủ yếu trên dữ liệu tiếng Anh.