

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC CMC**

**CÔNG TRÌNH THAM DỰ**

**HỘI NGHỊ SINH VIÊN NGHIÊN CỨU KHOA HỌC CẤP KHOA**

NĂM HỌC 2024 – 2025

Tên công trình: **Ứng dụng học sâu trong bài toán dự báo giá cổ**

**phiếu cho nhiều công ty dựa trên dữ liệu chuỗi thời gian**

Nhóm sinh viên thực hiện:

|  |  |  |
| --- | --- | --- |
| 1. Lê Hải Đức, | mã SV: BCS230118, | lớp: 23IT-GM |
| 2. Trần Quỳnh Anh, | mã SV: BIT230008, | lớp: 23IT-GM |

Đơn vị: Khoa CNTT - TT

|  |  |
| --- | --- |
| Giảng viên hướng dẫn: | TS. Phạm Thị Kim Dung  TS. Ngô Hoàng Huy |
|  |  |

***Hà Nội, tháng 04 năm 2025***

**1**

# TÓM TẮT

Dự báo giá cổ phiếu là một công nghệ tiên tiến, đang đóng vai trò ngày càng quan trọng tại Việt Nam và thế giới. Nghiên cứu này tập trung vào việc áp dụng mô hình CorrSTN vốn chỉ được sử dụng cho bài toán dự báo lưu lượng giao thông vào bài toán dự báo giá cổ phiếu cho nhiều công ty do có sự tương đồng về dữ liệu, đặc biệt chú trọng đến độ tương quan về không thời gian và khả năng trích xuất các đặc trưng quan trọng sử dụng cơ chế Attention. Bộ dữ liệu thực nghiệm là dữ liệu giá cổ phiếu của 10 công ty thuộc về 3 lĩnh vực là công nghệ, kinh tế và tiêu dùng được lấy từ 1/1/2012 tới 12/12/2024 và bao gồm 3.257 bản ghi theo ngày cho mỗi công ty, sau đó kết quả được đánh giá dựa trên các chỉ số hiệu suất chính như độ chính xác về giá đóng cửa của các công ty cho những ngày trong tương lai. Kết quả cho thấy rằng mô hình mang lại những cải thiện đáng kể trong đó tỉ lệ sai số là 1.42%. Nghiên cứu này đề xuất tiếp tục cải tiến mô hình và khám phá các giải pháp kỹ thuật tiên tiến hơn để tối ưu hóa hiệu suất, đáp ứng tốt hơn sự phức tạp và đa dạng của dữ liệu giá cổ phiếu trong thực tế.

**Từ khóa:** CorrSTN

Chúng tôi xin cam đoan công trình nghiên cứu là do nhóm thực hiện. Tất cả những tham khảo từ các nghiên cứu liên quan đều được nêu nguồn gốc một cách rõ ràng, minh bạch. Các chương trình thực nghiệm đều là do nhóm thực hiện mà có được, không sao chép từ bất cứ nguồn nào.

## Dẫn nhập

Trong thời đại công nghệ số, nhận dạng giọng nói tự động (ASR) đã trở thành một công nghệ chủ lực trong nhiều ứng dụng, từ trợ lý ảo, dịch vụ chăm sóc khách hàng, đến hỗ trợ người khiếm thính. Việc phát triển các hệ thống ASR không chỉ mang lại sự tiện lợi trong giao tiếp mà còn mở ra tiềm năng lớn trong việc nâng cao hiệu quả và trải nghiệm người dùng. Tuy nhiên, quá trình xây dựng mô hình ASR cho tiếng Việt gặp phải nhiều thách thức riêng, đặc biệt là về ngữ âm và thanh điệu. Với sự khác biệt rõ rệt giữa các vùng miền cùng với hệ thống 6 thanh điệu phức tạp, tiếng Việt yêu cầu các mô hình ASR có khả năng xử lý và nhận diện hiệu quả những biến thể âm thanh này.

Hơn nữa, sự hạn chế về dữ liệu tiếng Việt cũng đặt ra bài toán khó khăn trong việc huấn luyện các mô hình ASR với độ chính xác cao. So với các ngôn ngữ khác, tài nguyên dữ liệu dành cho tiếng Việt còn khá hạn chế, gây ảnh hưởng đến quá trình tối ưu hóa và huấn luyện mô hình. Chính vì vậy, việc thu thập, làm giàu dữ liệu và áp dụng các kỹ thuật tiên tiến như học sâu, Transformer, và Conformer trở thành những giải pháp cốt lõi để giải quyết những thách thức này.

Nghiên cứu này tập trung vào việc phát triển các mô hình ASR tiên tiến cho tiếng Việt, cụ thể là Wav2Vec2 và Fast-Conformer. Chúng tôi sẽ phân tích quy trình thu thập, xử lý dữ liệu và đánh giá hiệu quả của từng mô hình thông qua các thử nghiệm thực tế. Nghiên cứu hướng đến việc tối ưu hóa độ chính xác, giảm tỷ lệ lỗi từ và đề xuất các cải tiến nhằm nâng cao hiệu suất nhận dạng giọng nói tiếng Việt trong những ứng dụng thực tế.

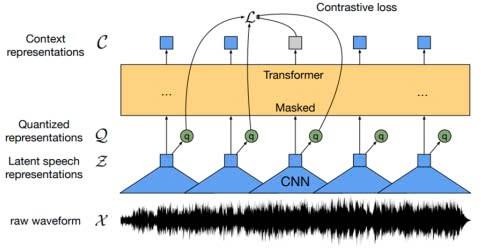
## Một số mô hình nhận dạng giọng nói

Trong quá trình phát triển hệ thống nhận diện giọng nói tự động (ASR) tiếng Việt, việc lựa chọn mô hình là một bước quan trọng quyết định đến độ chính xác và hiệu suất của hệ thống. Hai mô hình tiên tiến, Wav2vec và Fast Conformer, được lựa chọn vì khả năng vượt trội trong việc xử lý và nhận dạng ngôn ngữ từ tín hiệu âm thanh.

Wav2vec nổi bật với khả năng học biểu diễn không cần nhãn, giúp khai thác tối đa dữ liệu âm thanh thô, trong khi Fast Conformer kết hợp giữa sự mạnh mẽ của Transformer và hiệu quả xử lý nhanh của Conformer. Chính vì vậy, hai mô hình này được lựa chọn để tối ưu hóa hệ thống ASR tiếng Việt trong dự án hiện tại.

### Wav2vec2

Wav2Vec2 là một mô hình học sâu tiên tiến được phát triển bởi Facebook AI Research (FAIR), được thiết kế chủ yếu để xử lý âm thanh và nhận diện giọng nói. Mô hình dựa trên kiến trúc Transformer và nổi bật với khả năng học tự giám sát từ dữ liệu âm thanh thô mà không cần nhãn, giúp giảm đáng kể nhu cầu về dữ liệu có nhãn trong các bài toán nhận diện giọng nói.



**Hình 1:** Kiến trúc mô hình wav2vec2 [1]

**Feature Encoder**: Tín hiệu âm thanh thô được xử lý qua các bước chia nhỏ, chuẩn hóa, và cắt khung để chuẩn bị cho việc trích xuất đặc trưng. Sau đó, tín hiệu âm thanh được xử lý qua các lớp convolution nhằm trích xuất các đặc trưng âm thanh quan trọng. Kết quả của quá trình này là một chuỗi các vector đặc trưng, phản ánh thông tin về tần số và biên độ của âm thanh.

**Context Network**: Các vector đặc trưng được đưa vào một mạng Transformer, nơi cơ chế tự chú ý (self-attention) được áp dụng để học ngữ cảnh dài hạn trong chuỗi dữ liệu. Phương pháp này giúp cải thiện khả năng nhận diện giọng nói, đặc biệt trong các ngữ cảnh phức tạp, bằng cách hiểu rõ hơn các mối quan hệ và sự phụ thuộc giữa các phần của dữ liệu âm thanh.

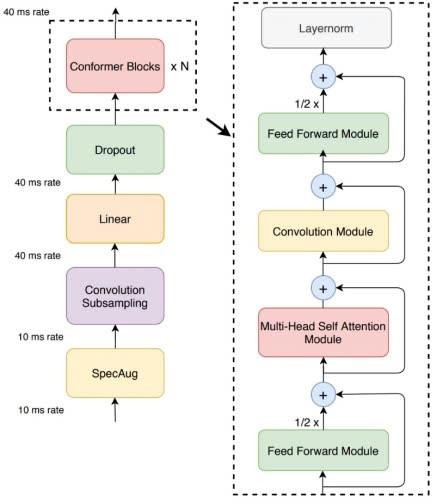
**Quantization Module**: Trong giai đoạn này, các vector đặc trưng liên tục được chuyển đổi thành các biểu diễn rời rạc thông qua kỹ thuật lượng tử hóa. Quá trình này giúp mô hình học các mẫu cấu trúc trong dữ liệu âm thanh một cách hiệu quả hơn, đặc biệt là trong huấn luyện không nhãn. Việc tạo ra các biểu diễn rời rạc từ các biểu diễn liên tục giúp mô hình nhận diện và học các mẫu cấu trúc trong dữ liệu âm thanh dễ dàng hơn.

### Fast-Conformer

**Convolution-augmented Transformer (Conformer)**: Conformer là sự kết hợp tiên tiến giữa hai loại mô hình học sâu nổi bật: Transformer và Convolutional Neural Networks (CNNs). Mô hình này kết hợp sức mạnh của CNNs trong việc trích xuất đặc

trưng cục bộ với khả năng của Transformers trong việc hiểu ngữ cảnh dài hạn và toàn cục. Kết quả là Conformer đạt được độ chính xác rất cao trong các tác vụ liên quan đến dữ liệu chuỗi, chẳng hạn như nhận diện giọng nói và phân tích văn bản.

Sự kết hợp giữa Convolution và Transformer: Convolutional Neural Networks (CNNs): CNNs nổi bật trong việc nắm bắt các đặc trưng cục bộ của dữ liệu. Trong các ứng dụng như âm thanh và văn bản, các khối convolution giúp mô hình trích xuất các mẫu ngắn hạn một cách hiệu quả. Ví dụ, trong nhận diện giọng nói, CNNs có thể nhận diện các âm tiết hoặc từ ngữ từ tín hiệu âm thanh. Ngược lại, Transformers sử dụng cơ chế self-attention để nắm bắt các mối quan hệ dài hạn và sự phụ thuộc toàn cục trong chuỗi dữ liệu. Điều này giúp mô hình hiểu ngữ cảnh rộng hơn, như câu hoặc đoạn văn, và làm cho mô hình trở nên mạnh mẽ trong việc xử lý các thông tin phức tạp và dài hạn. Nhờ vào các cải tiến trong việc kết hợp convolution và attention, Conformer khai thác cả đặc trưng địa phương và toàn cục trong dữ liệu, có khả năng nắm bắt các đặc điểm ngữ âm và ngữ nghĩa tốt hơn, đặc biệt trong các ngữ cảnh phức tạp.



**Hình 2:** Kiến trúc mô hình fast conformer [2]

**Fast-Conformer** là phiên bản tối ưu hóa của Conformer, nhằm cải thiện hiệu suất và tốc độ tính toán. Mặc dù Fast-Conformer duy trì những lợi ích của Conformer trong việc kết hợp CNN và Transformer, nhưng nó được thiết kế để giảm thiểu độ phức tạp tính toán và tăng tốc quá trình xử lý. Điều này giúp Fast-Conformer trở thành một lựa chọn lý tưởng cho các ứng dụng yêu cầu xử lý nhanh chóng và hiệu quả.

### So sánh kiến trúc hai mô hình Wav2Vec2 và Fast Conformer

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **Wav2Vec2** | **Fast Conformer** |
| Loại mô hình | **Self-Supervised Learning (SSL)**: Mô hình dựa trên học không giám | **Hybrid Model (Conformer)**: Sự kết hợp giữa Transformer và Convolution, |

|  |  |  |
| --- | --- | --- |
|  | sát, được huấn luyện với dữ liệu không gán nhãn. | tối ưu hóa cho xử lý ngôn ngữ tự nhiên trong thời gian thực. |
| Kiến trúc tổng thể | **Transformer Encoder**: Toàn bộ kiến trúc dựa trên Transformer với multi- head self-attention để học các đặc trưng ngữ cảnh dài từ dữ liệu thô. | **Conformer (Conv + Self-Attention)**: Conformer kết hợp Transformer (self- attention) cho ngữ cảnh dài và Convolution để nắm bắt thông tin cục bộ. |
| Input đầu vào | **Waveform thô**: Mô hình làm việc trực tiếp trên sóng âm thanh thô, không cần tiền xử lý phức tạp như MFCC hoặc spectrogram. | **Waveform hoặc Feature Extracted**: Có thể nhận dữ liệu thô hoặc đã qua xử lý, sử dụng CNN để trích xuất các đặc điểm cục bộ trong tín hiệu. |
| Cơ chế học | **Contrastive Learning**: Học qua đối lập (positive và negative samples) để tạo các biểu diễn ẩn phong phú từ dữ liệu không gán nhãn. | **Hybrid Attention Mechanism**: Sử dụng self-attention để học ngữ cảnh toàn cầu, kết hợp với Convolution để học thông tin cục bộ của tín hiệu. |
| Biến đổi tín hiệu âm thanh | Sử dụng **quantization** để biến đổi tín hiệu âm thanh thành các mã nhị phân. | Dùng **Positional Encoding** trong Conformer và cơ chế **Local Convolutions** để nắm bắt thông tin thời gian. |
| Self-Attention (Attention Cơ bản) | **Multi-Head Self-Attention**: Tính toán mối quan hệ giữa các phần tử trong chuỗi bằng self-attention, rất mạnh trong mô hình hóa chuỗi dài. | **Efficient Scaled Self-Attention**: Conformer giảm độ phức tạp của self- attention để tăng tốc độ và giữ được thông tin ngữ cảnh hiệu quả. |
| Convolutional Layers | **Không sử dụng**: Wav2Vec2 chỉ dựa trên self-attention, không có thành phần convolution. | **Depthwise Separable Convolution**: Conformer sử dụng các convolution layers để học các đặc điểm cục bộ như biến đổi âm vực và tốc độ nói. |
| Positional Encoding | **Learnable Positional Embedding**: Sử dụng các embedding học được để biểu diễn vị trí của các từ trong chuỗi âm thanh. | **Relative Positional Encoding**: Tối ưu hóa theo thứ tự tương đối thay vì tuyệt đối, giúp cải thiện tính chính xác khi làm việc với chuỗi dài. |
| Xử lý ngữ cảnh (Context Modeling) | **Global Contextual Modeling**: Dựa vào self-attention, học các mối quan | **Hybrid Context Modeling**: Kết hợp giữa self-attention và convolution, hiệu |

|  |  |  |
| --- | --- | --- |
|  | hệ trên toàn chuỗi giọng nói dài, tốt cho các ngữ cảnh phức tạp. | quả trong việc nắm bắt thông tin ngữ cảnh và cục bộ. |
| Ứng dụng | **Tập trung vào học không giám sát**: Thích hợp cho các dự án cần tận dụng lượng lớn dữ liệu không gán nhãn. | **Thích hợp cho thời gian thực**: Phù hợp với các ứng dụng yêu cầu thời gian suy luận nhanh và tài nguyên tính toán hạn chế. |
| Các ứng dụng chính | Nhận dạng giọng nói không gán nhãn, tạo embedding giọng nói, trích xuất đặc trưng từ dữ liệu âm thanh thô. | Nhận dạng giọng nói thời gian thực, dịch vụ thoại trực tiếp, tối ưu hóa cho các thiết bị di động và nhúng. |

**Bảng 1:** Bảng so sánh kiến trúc 2 mô hình Wav2Vec2 và Fast-Conformer

## Thực nghiệm

### Bài toán thực nghiệm

**Mục tiêu của thực nghiệm**: Mục tiêu của thực nghiệm là nhận dang giọng nói sử dụng hai mô hình Wav2vec và Fast Conformer và so sánh hiệu suất của hai mô hình trong bài toán nhận diện giọng nói tự động (ASR) cho tiếng Việt. Thực nghiệm này nhằm tìm ra giải pháp tối ưu cho việc chuyển đổi âm thanh thành văn bản, với trọng tâm là xử lý chính xác các đoạn hội thoại tiếng Việt. Điều này bao gồm việc phản ánh đầy đủ sự phong phú về ngữ điệu và giọng vùng miền trong tiếng Việt.

**Dữ liệu đầu vào**: Là các file âm thanh dạng wav ghi lại âm thanh giọng nói tiếng Việt từ nhiều vùng miền khác nhau. Những đoạn âm thanh này chứa các giọng nói tự nhiên với sự biến đổi về phát âm và ngữ điệu, phản ánh sự đa dạng trong cách nói và biểu cảm của người nói. Sự đa dạng này tạo ra thách thức trong việc nhận diện và chuyển đổi chính xác các đoạn hội thoại.

**Dữ liệu đầu ra**: Là file văn bản dạng text chuyển đổi từ bản ghi âm thanh tương ứng, với mục tiêu phản ánh chính xác nội dung của âm thanh. Văn bản chuyển đổi cần đảm bảo độ chính xác cao trong việc nhận diện từ ngữ và cấu trúc câu, nhằm đảm bảo rằng thông tin được truyền đạt một cách chính xác và rõ ràng.

Thông qua thực nghiệm, nghiên cứu sẽ cung cấp cái nhìn rõ ràng về khả năng của mỗi mô hình trong việc xử lý ngôn ngữ tiếng Việt. Mục tiêu là tìm ra mô hình hoặc phương

pháp tối ưu nhất cho hệ thống ASR tiếng Việt trong dự án, giúp cải thiện hiệu suất và độ chính xác của việc chuyển đổi âm thanh thành văn bản trong tiếng Việt.

### Môi trường thực nghiệm

Quá trình triển khai thực nghiệm được thực thi trên Kaggle Notebook, sử dụng GPU P100 ổn định, liên kết đến Hugging Face để load dataset, model và theo dõi quá trình huấn luyện trực quan trên Wandb. Cài đặt các thư viện liên quan như nemo\_toolkit, torch, audiomentations, librosa, tqdm để xử lý dữ liệu, huấn luyện mô hình.

### Dữ liệu thực nghiệm

#### Tele Dataset

Data Tele được cung cấp bởi Viện ATI, là một tập hợp dữ liệu tiếng Việt đa dạng, phục vụ cho nghiên cứu trong lĩnh vực nhận diện giọng nói tự động (ASR) với trọng tâm là các ứng dụng thực tiễn trong dịch vụ chăm sóc khách hàng. Dữ liệu bao gồm các bản ghi âm từ các tình huống thực tế trong nhiều lĩnh vực như bảo hiểm, viễn thông, tài chính và dịch vụ ô tô. Đây là một bộ dữ liệu có tính đặc thù cao, đại diện cho các cuộc hội thoại chuyên nghiệp trong môi trường công việc.



**Hình 3:** Nội dung tập dữ liệu Tele

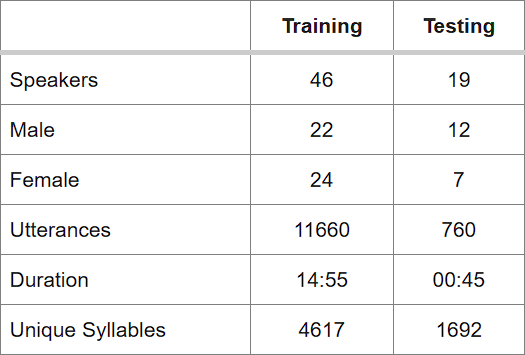
**Đặc điểm của tập dữ liệu**: Tập dữ liệu có khoảng 12.000 bản ghi với dung lượng khoảng 10 giờ dữ liệu. Trong đó, có bản ghi chứa nhiều từ là tên riêng bằng tiếng Anh, hoặc một số ký tự đặc biệt. Điều này làm phong phú thêm nội dung, đồng thời làm tăng độ phức tạp trong việc nhận diện và độ chính xác của mô hình.

Tập dữ liệu được thu thập từ các tình huống thực tế trong công việc, chứa nhiều tiếng ồn, nhiễu, và chất lượng âm thanh không tốt. Điều này phản ánh sự đa dạng và thách thức của môi trường thu âm thực tế. Do đây là một bộ dữ liệu thực tế với ngữ cảnh nghiệp vụ riêng, việc triển khai huấn luyện mô hình sẽ không yêu cầu thực hiện các phương pháp tăng cường dữ liệu. Bộ dữ liệu đã đủ đa dạng và đại diện cho các tình huống thực tế, giúp

mô hình học tập và thích ứng với các yếu tố nhiễu và biến đổi trong môi trường công việc thực tế.

#### Vivos

VIVOS là một kho ngữ liệu tiếng Việt miễn phí gồm 15 giờ ghi âm giọng nói được chuẩn bị cho nhiệm vụ Nhận dạng giọng nói tự động (ASR).Kho ngữ liệu được chuẩn bị bởi AILAB của Đại học Khoa học Tự nhiên – ĐHQGHCM, được cung cấp trên Hugging Face: [https://huggingface.co/datasets/AILAB-VNUHCM/Vivos](https://huggingface.co/datasets/AILAB-VNUHCM/vivos)



**Hình 4:** Nội dung tập dữ liệu Vivos

**Đặc điểm của tập dữ liệu**: Tập dữ liệu được ghi âm trong một không gian yên tĩnh, với sự trang bị micro chất lượng cao, và được thu thập từ các người nói được yêu cầu đọc từng câu một cách rõ ràng. Do đó, dữ liệu này được coi là sạch và không bị nhiễu, tạo điều kiện thuận lợi cho việc huấn luyện mô hình nhận diện giọng nói.

Tập dữ liệu chứa gần 13.000 dòng dữ liệu với sự đa dạng về người nói và giới tính. Điều này giúp mô hình học được các đặc điểm khác nhau của giọng nói từ nhiều nguồn khác nhau. Trong transcript của dữ liệu, có nhiều ký tự đặc biệt xuất hiện, làm phong phú thêm nội dung và tăng tính thách thức trong việc xử lý văn bản.

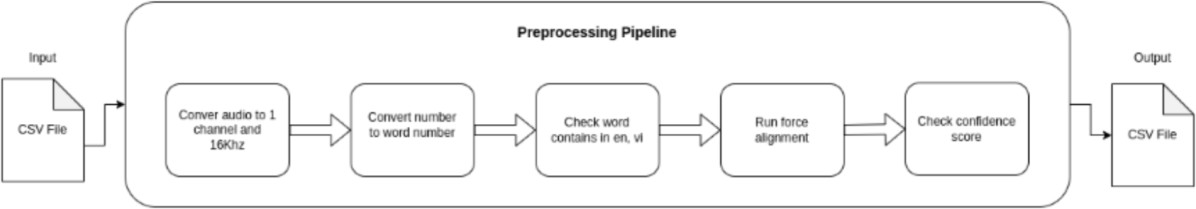
Mặc dù đây là một tập dữ liệu sạch với sự đa dạng về ngữ cảnh và người nói, việc tăng cường dữ liệu âm thanh trong quá trình huấn luyện là cần thiết để làm cho mô hình

tổng quát hóa tốt hơn và có khả năng hoạt động hiệu quả trong môi trường thực tế. Phương pháp này sẽ giúp mô hình thích nghi với các tình huống có sự biến động về chất lượng âm thanh hoặc điều kiện ghi âm, từ đó cải thiện khả năng nhận diện chính xác trong các điều kiện phức tạp hơn.

### Thu thập và tiền xử lý dữ liệu

Tải bộ dữ liệu mong muốn về từ Hugging Face, lưu trữ trên bộ nhớ Kaggle Notebook.

Quy trình tiền xử lý dữ liệu được mô tả như hình 5.



**Hình 5:** Quy trình tiền xử lý dữ liệu [3]

Xác định được các giá trị dữ liệu ngoại lệ, các thuộc tính dữ liệu không cần thiết để loại bỏ và chia thành các tập dữ liệu phù hợp với tỉ lệ train – validation – test là 80-10-10.

Đảm bảo dữ liệu transcript được chuẩn hóa: chuyển số thành chữ, xóa các khoảng trắng thừa và các kí tự đặc biệt, chuyển về dạng chữ thường để tối ưu trong quá trình huấn luyện mô hình.

Đối với các tập dữ liệu sạch như Vivos, việc áp dụng các phương pháp tăng cường dữ liệu như chèn nhiễu, cao độ,… mức độ phù hợp cũng cần được thực hiện để tạo được bộ dữ liệu phù hợp với thực tế.

### Mã hóa dữ liệu đầu vào

**Tạo tokenizer và bộ từ vựng:** Để mã hóa dữ liệu văn bản cho mô hình, chúng tôi cần tạo một tokenizer và bộ từ vựng. Việc tạo tokenizer và bộ từ vựng là bước quan trọng trong quá trình chuẩn bị dữ liệu. Điều này giúp chuyển đổi văn bản thành dạng mà mô hình học máy có thể hiểu và học từ đó, đồng thời cải thiện hiệu suất và khả năng tổng quát của mô hình.

Đối với mô hình Wav2Vec2, chúng tôi sử dụng phương pháp mã hóa theo ký tự với kỹ thuật Sentenpiece Encoding, chọn mã hóa là ‘char’ (spe\_type = ‘char’). Điều này có nghĩa là mỗi ký tự trong văn bản sẽ được mã hóa riêng biệt, và bộ từ vựng sẽ bao gồm tất cả các ký tự xuất hiện trong tập dữ liệu. Phương pháp này giúp mô hình nhận diện chính xác từng ký tự trong văn bản.

Đối với mô hình Fast-Conformer, chúng tôi sử dụng kỹ thuật Sentenpiece Encoding theo phương pháp Byte-Pair Encoding (BPE) với spe\_type = ‘bpe’. Phương pháp này chia văn bản thành các tiểu từ (subwords) dựa trên thuật toán BPE, giúp giảm số lượng từ trong bộ từ vựng và cải thiện khả năng tổng quát của mô hình. Bằng cách này, mô hình có thể học được các cấu trúc từ vựng phổ biến và ít phổ biến một cách hiệu quả hơn.

**Chuẩn hóa dữ liệu âm thanh**: thực hiện hai bước chính: chuẩn hóa độ dài dữ liệu và chuyển đổi âm thanh thành Tensor. Đối với việc chuẩn hóa độ dài, phân bổ miền âm thanh và chuẩn hóa độ dài âm thanh không quá 20s. Nếu đoạn âm thanh quá dài, chúng tôi cắt bớt để đạt được độ dài mong muốn. Ngược lại, nếu đoạn âm thanh quá ngắn, chúng tôi chèn thêm âm thanh trắng (silence) để đạt được độ dài phù hợp.

Sau khi chuẩn hóa độ dài, dữ liệu âm thanh sẽ được chuyển đổi thành Tensor bằng cách sử dụng Pytorch. Tensor là cấu trúc dữ liệu chính trong các thư viện học máy, và việc chuyển đổi âm thanh thành Tensor cho phép lưu trữ và xử lý dữ liệu một cách hiệu quả trong quá trình huấn luyện mô hình.

### Xây dựng mô hình

Với dung lượng các tập dữ liệu được sử dụng có dung lượng khá lớn (khoảng 10 – 30 giờ dữ liệu âm thanh), cần thiết lập mô hình với dung lượng phù hợp: ở đây ta sử dụng mô hình Wav2Vec2-base và Fast-Conformer CTC Large để triển khai bài toán và áp dụng bộ mã hóa Tokenizer trước đó vào mô hình tương ứng để đạt hiệu quả.

Thiết lập tham số huấn luyện: Xác định các tham số như tốc độ học, kích thước batch, số lượng epoch, điều chỉnh giảm trọng số, và các tham số khác để tối ưu hóa quá trình huấn luyện.

Thiết lập bộ huấn luyện (Trainer): Sử dụng thư viện như Hugging Face Transformers để cấu hình và khởi tạo bộ huấn luyện, quản lý và theo dõi quá trình huấn luyện mô hình trên Wandb.

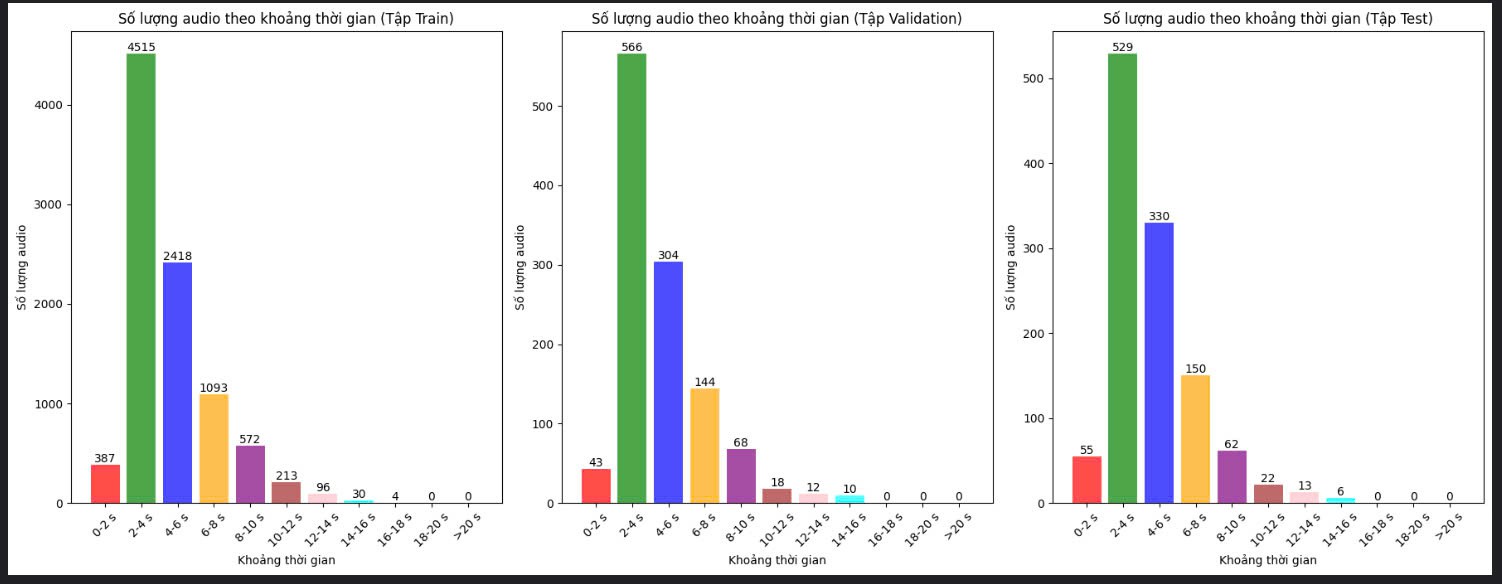
## Kết quả thực nghiệm

### Wav2Vec2

Chúng tôi thực nghiệm mô hình Wav2Vec2 trên tập dữ liệu Vivos, với dung lượng các tập dữ liệu khi được chia train – validation – test như Bảng 2, và thống kê số lượng các mẫu audio theo thời lượng file như Hình 6 dưới đây.

|  |  |  |
| --- | --- | --- |
| **Dataset** | **Số lượng mẫu** | **Dung lượng (Giờ)** |
| Train | 9328 | 11.9 |
| Validation | 1165 | 1.51 |
| Test | 1167 | 1.51 |

**Bảng 2:** Dung lượng tập dữ liệu Vivos



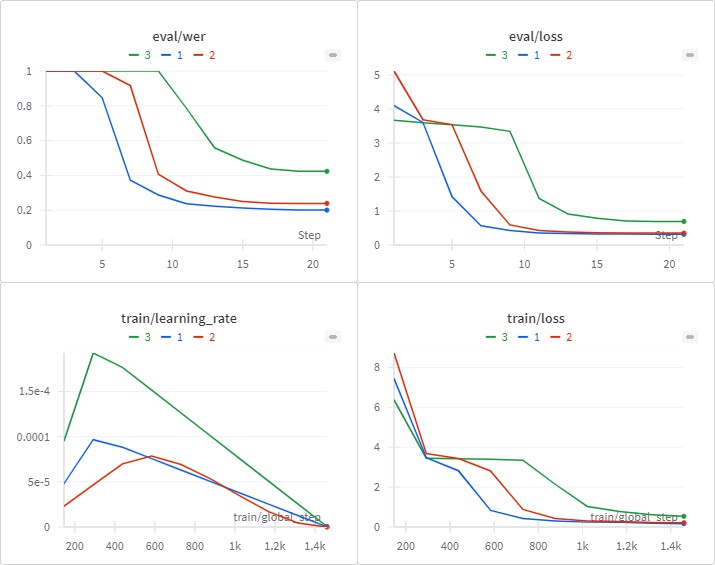
**Hình 6:** Thống kê số lượng bản ghi Vivos theo thời lượng Bảng kết quả các lần thử nghiệm

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Dataset | Tham số | | Kết quả | | | |
| Train\_loss | Val\_loss | Val\_Wer | Test\_Wer |
| 1 | Vivos | Lr Warmup\_steps  Lr\_scheduler\_type  Epochs | 1e-4  300  linear 20 | 0.1664 | 0.315 | 0.2 | 0.1935 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Total steps | 1460 |  |  |  |  |
| 2 | Vivos | Lr Warmup\_steps  Lr\_scheduler\_type Epochs  Total steps | 8e-5  500  cosine 20  1460 | 0.211 | 0.3501 | 0.2392 | 0.2 |
| 3 | Augmented Vivos | Lr Warmup\_steps  Lr\_scheduler\_type Epochs  Total steps | 2e-4  300  linear 20  1460 | 0.5388 | 0.6926 | 0.4232 | 0.4305 |

**Bảng 3:** Kết quả thử nghiệm tập dữ liệu Vivos

Chúng tôi đã trực quan hóa quá trình huấn luyện các lần thử như Hình 7.



**Hình 7:** Trực quan hóa kết quả thử nghiệm Vivos

Kết quả các lần chạy thực nghiệm trên tập dữ liệu Vivos (Bảng 3) được phân tích cụ thể như sau**:**

* **Lần 1**: Trong thí nghiệm đầu tiên, mô hình đạt được kết quả **Train\_loss** khá thấp ở mức 0.1664, tuy nhiên, **Val\_loss** cao hơn đáng kể ở mức 0.315. Sự chênh lệch lớn giữa **Train\_loss** và **Val\_loss** có thể chỉ ra rằng mô hình đang bị overfitting với dữ liệu huấn luyện. Mặc dù Wer (Word Error Rate) trên tập validation và test là 0.2 và 0.1935, cho thấy khả năng nhận diện từ của mô hình là khá tốt, vẫn cần nỗ lực để giảm Wer hơn nữa nhằm cải thiện hiệu suất tổng quát của mô hình.
* **Lần 2**: Trong thí nghiệm thứ hai, việc giảm **learning rate** và tăng số bước warmup đã dẫn đến một mức Train\_loss cao hơn (0.211) so với thí nghiệm đầu tiên. Điều này cho thấy mô hình học chậm hơn và không đạt được kết quả tốt như thí nghiệm trước. **Val\_loss** và **Val\_Wer** cũng cao hơn (0.3501 và 0.2392 so với 0.315 và 0.2), cho thấy phương pháp sử dụng warmup steps cao hơn và **cosine scheduler** không mang lại hiệu quả tốt như **linear scheduler** trong trường hợp này.
* **Lần 3**: Trong thí nghiệm thứ ba, khi sử dụng dữ liệu tăng cường với **learning rate** lớn, **Train\_loss** khá cao ở mức 0.5388. Điều này cho thấy mô hình gặp khó khăn trong việc học từ dữ liệu tăng cường. **Val\_loss** và **Val\_Wer** cũng cao đáng kể (0.6926 và 0.4232), có thể chỉ ra rằng việc tăng cường dữ liệu chưa được tối ưu hoặc lượng noise trong dữ liệu tăng cường đã ảnh hưởng tiêu cực đến hiệu suất của mô hình.

Đánh giá**:**

* **Wav2Vec2**: Mô hình Wav2Vec2 hoạt động tương đối ổn định trên tập dữ liệu sạch như Vivos, với độ chính xác trung bình khoảng 80%. Để nâng cao hiệu suất, cần tăng dung lượng dữ liệu huấn luyện và thực hiện tinh chỉnh thêm nhằm tránh overfitting. Cần cân bằng giữa độ chính xác và khả năng tổng quát của mô hình để cải thiện kết quả trong các tình huống thực tế.
* **Hạn chế đối với dữ liệu tăng cường**: Mô hình Wav2Vec2 còn hạn chế khi làm việc với dữ liệu đã được tăng cường, đặc biệt là khi dữ liệu có nhiễu. Độ chính xác đạt được không cao, điều này cho thấy cần cải thiện phương pháp tăng cường dữ liệu để nâng cao hiệu suất mô hình trong các điều kiện thực tế.

### Fast-Conformer

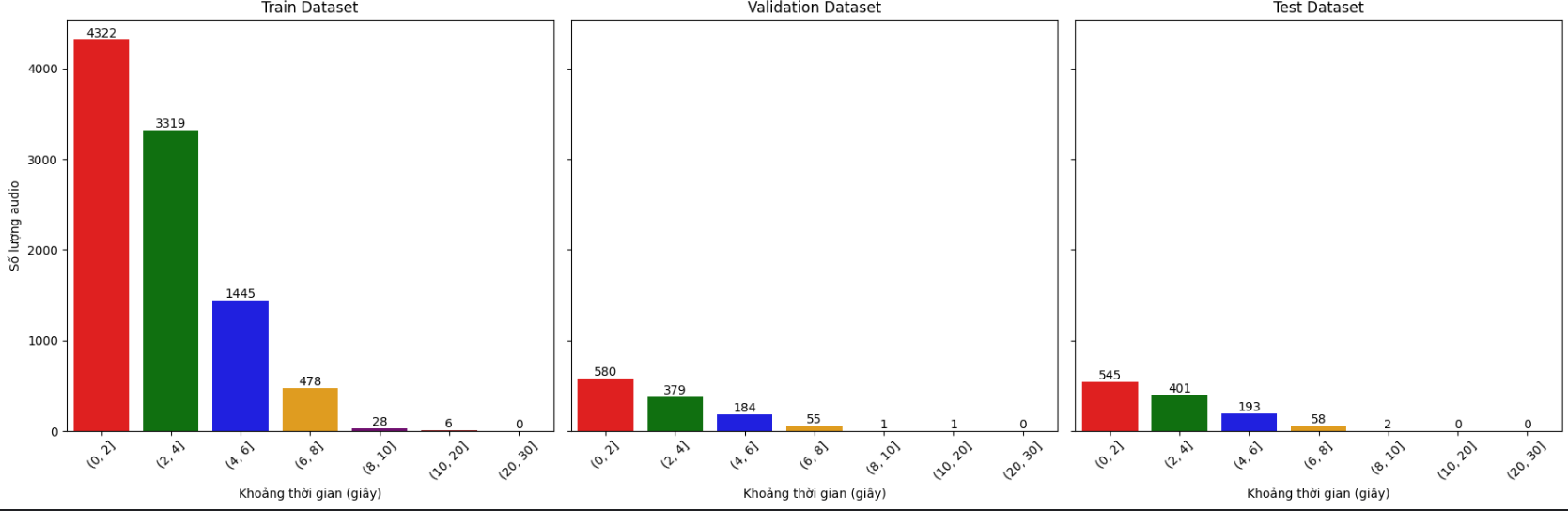
Triển khai bài toán ASR trên các dataset: **Tele**, **Vivos**, **Tele + Vivos**.

#### Tele Dataset

Chúng tôi thực nghiệm mô hình Fast-Conformer trên tập dữ liệu Tele, với dung lượng các tập dữ liệu khi được chia train – validation – test như Bảng 4, và thống kê số lượng các mẫu audio theo thời lượng file như Hình 8 dưới đây.

|  |  |  |
| --- | --- | --- |
| **Dataset** | **Số lượng mẫu** | **Dung lượng (Giờ)** |
| Train | 9598 | 7.35 |
| Validation | 1200 | 0.89 |
| Test | 1199 | 0.91 |

**Bảng 4:** Dung lượng tập dữ liệu Tele



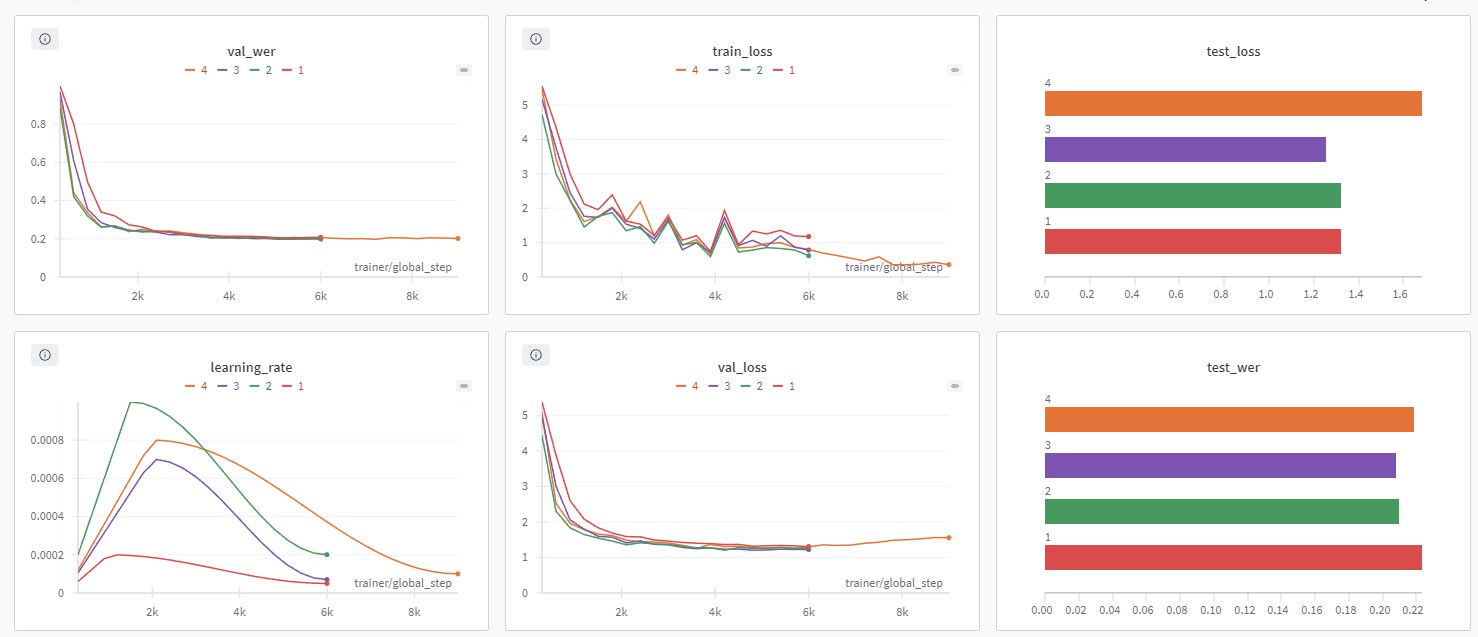
**Hình 8:** Thống kê số lượng bản ghi Tele theo thời lượng Bảng kết quả các lần thử nghiệm

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Dataset | Tham số | | Kết quả | | | | |
| Train\_loss | Val\_loss | Val\_Wer | Test\_loss | Test\_Wer |
| 1 | Tele | N\_layers Lr  Warmup\_steps Min\_lr Weight\_decay Epochs  Total steps | 18  2e-4  1000  5e-4  2e-4  20  6000 | 1.1754 | 1.3063 | 0.2048 | 1.3197 | 0.2233 |
| 2 | Tele | N\_layers  Lr | 18  1e-3 | 0.6188 | 1.265 | 0.2003 | 1.3233 | 0.2096 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Warmup\_steps Min\_lr Weight\_decay Epochs  Total steps | 1500  2e-4  1e-3  20  6000 |  |  |  |  |  |
| 3 | Tele | N\_layers Lr  Warmup\_steps Min\_lr Weight\_decay Epochs  Total steps | 18  7e-4  2000  7e-5  7e-4  20  6000 | 0.7845 | 1.2236 | 0.1992 | 1.2528 | 0.2077 |
| 4 | Tele | N\_layers Lr  Warmup\_steps Min\_lr Weight\_decay Epochs  Total steps | 19  8e-4  2000  8e-5  1e-4  30  9000 | 0.35796 | 1.55635 | 0.20167 | 1.68332 | 0.21882 |

**Bảng 5:** Kết quả thử nghiệm tập dữ liệu Tele

Chúng tôi đã trực quan hóa quá trình huấn luyện các lần thử như Hình 9:



**Hình 9:** Trực quan hóa kết quả thử nghiệm Tele

Kết quả các lần chạy thực nghiệm trên tập dữ liệu Tele (Bảng 5) được phân tích cụ thể như sau:

* **Lần 1**: Trong thử nghiệm đầu tiên, **Train\_loss** không thấp hơn **Val\_loss** nhiều, và **Test\_loss** tương đương với **Val\_loss**. Điều này cho thấy mô hình có khả năng tổng quát tương đối tốt khi áp dụng trên tập kiểm thử. Tuy nhiên, một điểm đáng lưu ý là **Test\_Wer** lại cao hơn so với **Val\_Wer**, điều này chỉ ra rằng khả năng dự đoán của mô hình trên tập kiểm thử chưa hoàn thiện và cần cải thiện thêm để tăng hiệu suất dự đoán.
* **Lần 2:** Trong trường hợp này, **learning rate** được tăng đáng kể, dẫn đến việc giảm mạnh **Train\_loss**. Tuy nhiên, sự chênh lệch giữa **Train\_loss** và **Val\_loss** cũng lớn hơn, cho thấy mô hình có thể đã bắt đầu xuất hiện hiện tượng **overfitting**. Dù vậy, **Test\_Wer** tương đương với **Val\_Wer**, điều này cho thấy mô hình dự đoán trên tập kiểm thử tốt hơn so với trường hợp 1, mặc dù có dấu hiệu **overfitting** nhẹ.
* **Lần 3:** Trong thử nghiệm này, **learning rate** ở mức trung bình giữa các thử nghiệm khác và số lượng **warmup steps** cao hơn, giúp đạt được kết quả tốt nhất về khả năng hội tụ và tổng quát hóa của mô hình. **Val\_loss** và **Test\_loss** giảm nhẹ, trong khi **Val\_Wer** và **Test\_Wer** giảm đáng kể so với các trường hợp trước. Điều này cho thấy khả năng dự đoán của mô hình trên cả tập **validation** và **kiểm thử** đã được cải thiện. Đây là trường hợp mà mô hình đạt được sự cân bằng tốt giữa việc tránh **overfitting** và khả năng dự đoán chính xác.
* **Lần 4:** Trong thử nghiệm cuối cùng, sử dụng số lượng **epochs** quá lớn không mang lại hiệu quả, dù **Train\_loss** giảm đáng kể nhưng **Val\_loss** và **Test\_loss** lại tăng cao. Điều này không chỉ không cải thiện độ chính xác của mô hình mà còn cho thấy hiện tượng **overfitting** rõ ràng. **Train\_loss** rất thấp, nhưng việc tăng số lượng **N\_layers** để làm cho mô hình phức tạp hơn không giúp cải thiện hiệu suất, mà ngược lại còn làm tăng sự chênh lệch giữa các chỉ số **loss**.

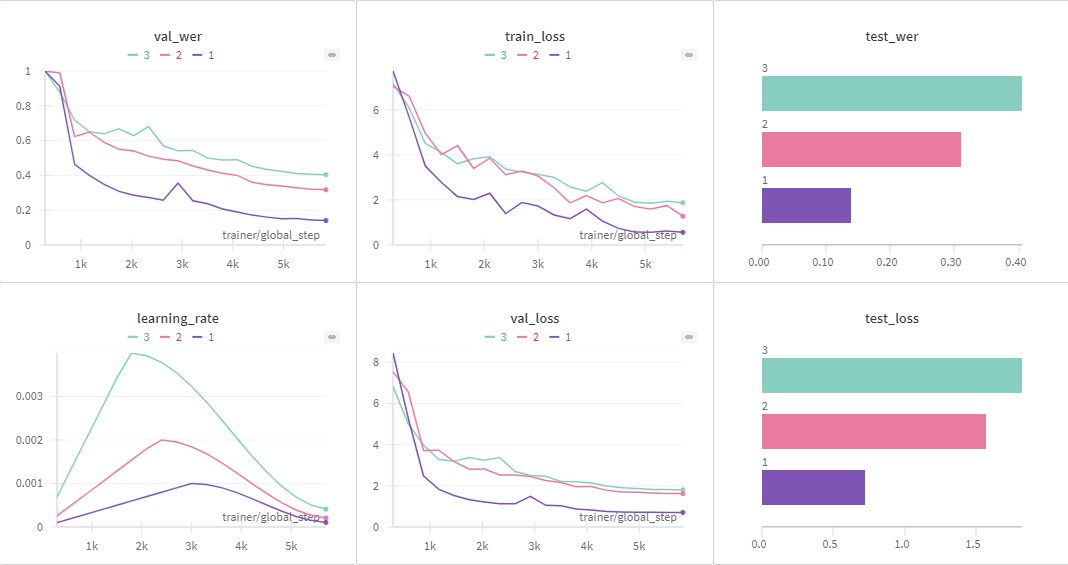
Đánh giá: Mô hình đạt sự ổn định tốt nhất trên tập dữ liệu **Tele** với ngữ cảnh trong lĩnh vực **Telesales** ở trường hợp 2, nơi **learning rate** hợp lý giúp đạt độ chính xác tốt mà không gây quá nhiều **overfitting**. Việc huấn luyện mô hình qua nhiều vòng lặp không cải thiện đáng kể độ chính xác và thậm chí còn làm tăng nguy cơ **overfitting**. Mô hình **Fast-Conformer** hoạt động khá tốt trên tập dữ liệu **Tele**, với độ chính xác trung bình khoảng 80%, và có thể được cải thiện thêm bằng cách tăng dung lượng bộ dữ liệu huấn luyện.

#### Vivos Dataset

Bảng kết quả các lần thử nghiệm:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Dataset | Tham số | | Kết quả | | | | |
| Train\_loss | Val\_loss | Val\_Wer | Test\_loss | Test\_Wer |
| 1 | Vivos | N\_layers Lr  Warmup\_steps Min\_lr Epochs  Total steps | 19  1e-3  3000  1e-4  20  6000 | 0.56225 | 0.70629 | 0.14047 | 0.72331 | 0.13808 |
| 2 | Augmented Vivos | N\_layers Lr  Warmup\_steps Min\_lr Epochs  Total steps | 19  2e-3  2000  2e-4  20  6000 | 1.27752 | 1.62282 | 0.31716 | 1.57248 | 0.31099 |
| 3 | Augmented Vivos | N\_layers Lr  Warmup\_ratio Min\_lr Epochs  Total steps | 19  4e-3  0.3  4e-4  20  6000 | 1.87302 | 1.80351 | 0.40351 | 1.82324 | 0.40533 |

**Bảng 6:** Kết quả thử nghiệm tập dữ liệu Vivos được làm giàu Chúng tôi đã trực quan hóa quá trình huấn luyện các lần thử như Hình 10:



**Hình 10:** Trực quan hóa kết quả thử nghiệm trên tập dữ liệu Vivos

Kết quả các lần chạy thực nghiệm trên tập dữ liệu Vivos (Bảng 6) được phân tích cụ thể như sau:

* + - * **Lần 1:** Trong thử nghiệm đầu tiên, mô hình cho thấy khả năng tổng quát hóa khá tốt trên tập dữ liệu Vivos sạch. Điều này được thể hiện qua việc không có sự chênh lệch lớn giữa **Train\_loss**, **Val\_loss**, và **Test\_loss**. Với tỉ lệ sai từ (**Wer**) chỉ ở mức 13,8%, mô hình đạt được sự cân bằng tốt giữa độ chính xác và khả năng tổng quát hóa. Đây là một kết quả khả quan khi làm việc với dữ liệu ít nhiễu và rõ ràng.
      * **Lần 2:** Ở thử nghiệm thứ hai, các chỉ số **loss** và **Wer** đều cao. Điều này cho thấy khi làm việc với dữ liệu phức tạp hơn, quá trình huấn luyện của mô hình gặp khó khăn hơn. Tuy nhiên, điểm tích cực là không có sự chênh lệch lớn giữa **Train\_loss**, **Val\_loss**, và **Test\_loss**, cho thấy mô hình vẫn có khả năng tổng quát hóa khá tốt ngay cả với dữ liệu Vivos được tăng cường. Đây có thể được xem là kết quả tốt nhất mà mô hình có thể đạt được trên tập dữ liệu này.
      * **Lần 3:** Trong thử nghiệm này, việc tăng tỉ lệ học không mang lại lợi ích mà thậm chí còn làm cho quá trình huấn luyện trở nên khó khăn hơn. Các chỉ số **loss** và **Wer** đều tăng cao, và **Train\_loss** còn cao hơn cả **Val\_loss** và **Test\_loss**, cho thấy

mô hình gặp khó khăn ngay cả trong quá trình huấn luyện. Điều này cho thấy rằng việc tăng tốc độ học có thể đã làm cho mô hình không kịp hội tụ và dẫn đến việc dự đoán kém chính xác.

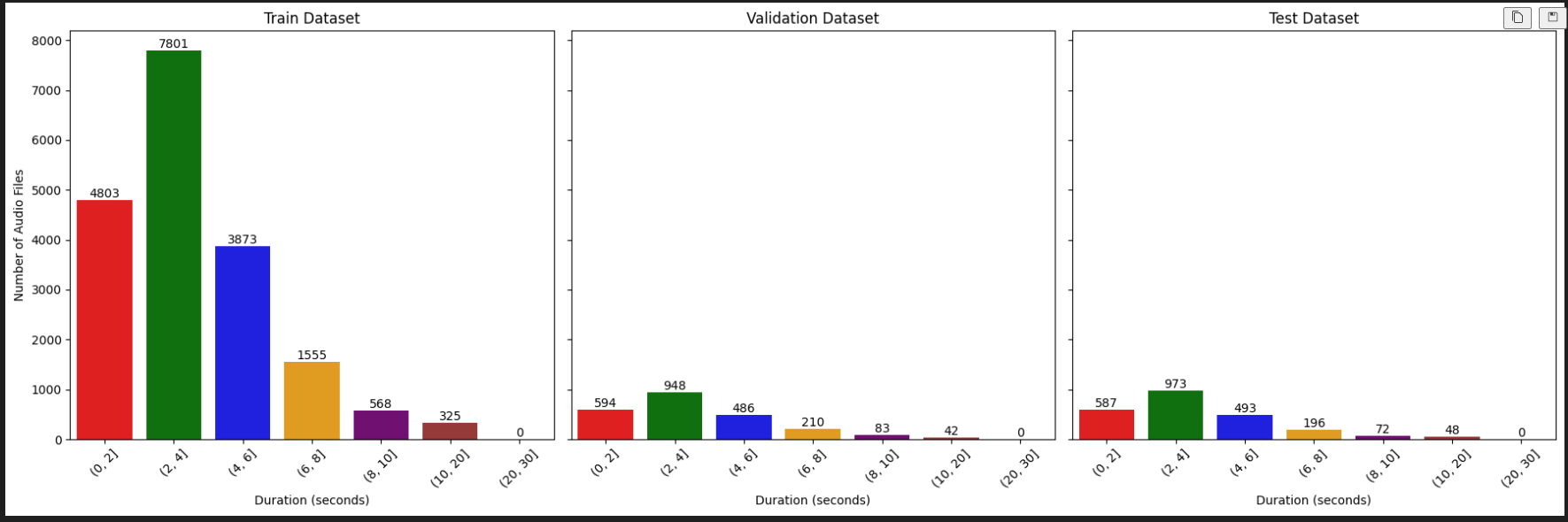
Đánh giá: Mô hình cho thấy khả năng tổng quát hóa tốt khi làm việc với tập dữ liệu có ngữ cảnh đa dạng và rõ ràng, không có nhiễu, với độ chính xác khoảng 86%. Tuy nhiên, khi làm việc với dữ liệu đã được tăng cường và có nhiễu, mô hình vẫn có thể xử lý, nhưng cần điều chỉnh tỉ lệ nhiễu một cách phù hợp hoặc tăng thêm dung lượng dữ liệu để đảm bảo rằng quá trình tăng cường dữ liệu không làm mô hình gặp khó khăn. Ngoài ra, việc tinh chỉnh các tham số huấn luyện, đặc biệt là tốc độ học và số bước warmup, là cần thiết để đảm bảo mô hình hoạt động hiệu quả hơn trên các loại dữ liệu khác nhau.

#### Tele + Vivos Dataset

Chúng tôi thực nghiệm mô hình Fast-Conformer trên tập dữ liệu kết hợp Tele và Vivos, với dung lượng các tập dữ liệu khi được chia train – validation – test như Bảng 7, và thống kê số lượng các mẫu audio theo thời lượng file như Hình 11 dưới đây.

|  |  |  |
| --- | --- | --- |
| **Dataset** | **Số lượng mẫu** | **Dung lượng (Giờ)** |
| Train | 18925 | 19.2 |
| Validation | 2363 | 2.46 |
| Test | 2369 | 2.42 |

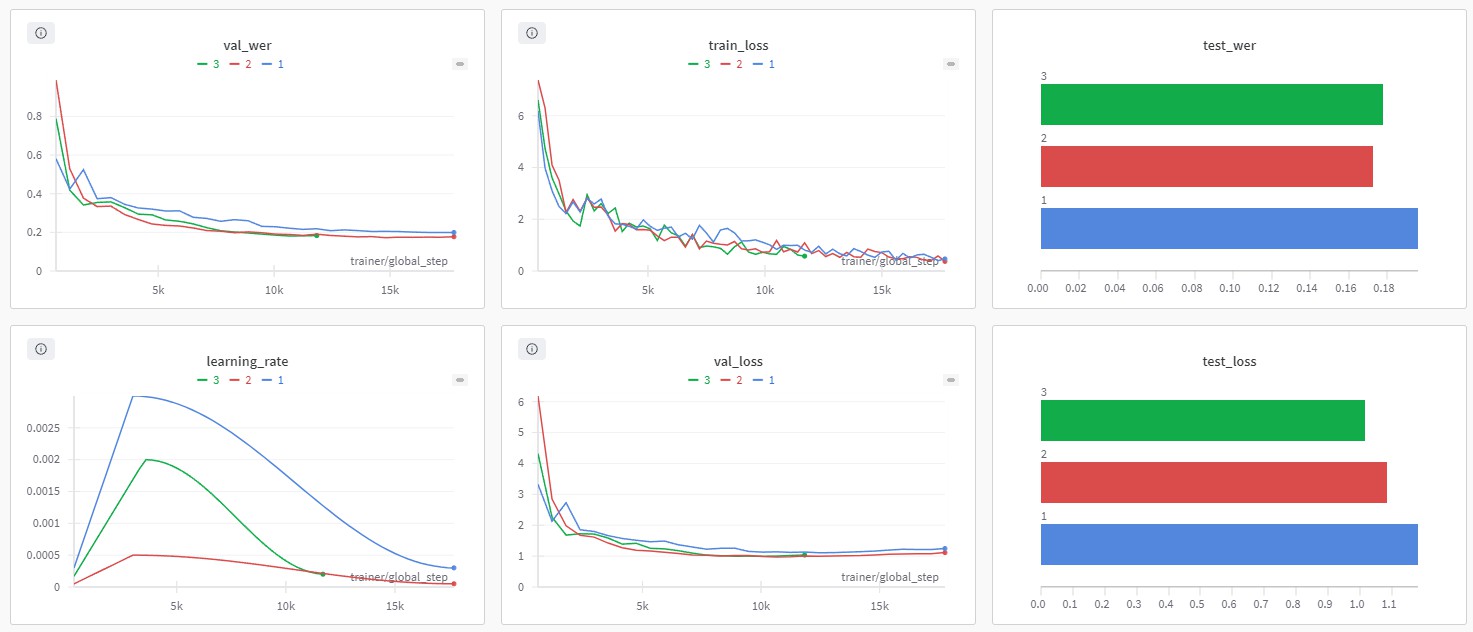
**Bảng 7:** Dung lượng tập dữ liệu Tele + Vivos



**Hình 11:** Thống kê số lượng bản ghi tele + Vivos theo thời lượng Bảng kết quả các lần thử nghiệm:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Dataset | Tham số | | Kết quả | | | | |
| Train\_loss | Val\_loss | Val\_Wer | Test\_loss | Test\_Wer |
| 1 | Tele + Vivos | N\_layers Lr  Warmup\_steps Min\_lr Weight\_decay Epochs  Total steps | 19  3e-3  3000  3e-3  3e-4  30  17760 | 0.46583 | 1.24575 | 0.19934 | 1.182 | 0.19594 |
| 2 | Tele + Vivos | N\_layers Lr  Warmup\_steps Min\_lr Weight\_decay Epochs  Total steps | 19  5e-4  3000  5e-4  5e-5  30  17760 | 0.37185 | 1.11381 | 0.17726 | 1.08634 | 0.17234 |
| 3 | Tele + Vivos | N\_layers Lr  Warmup\_ratio Min\_lr Weight\_decay Epochs  Total steps | 19  3e-4  0.3  3e-4  3e-5  20  11840 | 0.57483 | 1.03511 | 0.18337 | 1.0153 | 0.17753 |

**Bảng 8:** Kết quả thử nghiệm tập dữ liệu Vivos + tele Chúng tôi đã trực quan hóa quá trình huấn luyện các lần thử như Hình 12:



**Hình 12:** Trực quan hóa kết quả thử nghiệm trên tập dữ liệu Tele + Vivos

Kết quả các lần chạy thực nghiệm trên tập dữ liệu Vivos (Bảng 8) được phân tích cụ thể như sau:

* **Lần 1:** Trong thử nghiệm đầu tiên, mô hình sử dụng **tốc độ học** khá cao (3e-3) và số vòng lặp lớn (30), dẫn đến hiện tượng **overfitting**. Mặc dù **Train\_loss** đạt mức thấp, sự chênh lệch lớn giữa **Val\_loss** và **test\_loss** chỉ ra rằng mô hình gặp khó khăn trong việc tổng quát hóa trên tập kiểm thử. Dù **Wer** trên tập test đạt mức khá khả quan (19,6%), mô hình vẫn chưa hội tụ và chưa đạt được hiệu quả tốt nhất do tình trạng **overfitting**.
* **Lần 2**: Khi giảm sâu tốc độ học xuống (5e-4) và giữ nguyên số vòng lặp (30), các chỉ số **loss** giảm nhẹ, nhưng hiện tượng **overfitting** vẫn xuất hiện, khiến mô hình chưa tổng quát tốt. **Wer** đã cải thiện, giảm xuống còn 17,2% trên tập test. Tuy nhiên, giống như trường hợp 1, mô hình vẫn chưa đạt được sự hội tụ tốt nhất.
* **Lần 3:** Với việc tăng tốc độ học lên (3e-4) và giảm số vòng lặp xuống 20, **Train\_loss** có tăng nhẹ, nhưng **Val\_loss** và **test\_loss** tiếp tục giảm. Điều này cho thấy mô hình dần khắc phục hiện tượng **overfitting** và đạt được khả năng tổng quát hóa tốt hơn. **Wer** trên tập test tăng nhẹ lên 17,7%, nhưng điều này phản ánh sự cân bằng tích cực hơn giữa khả năng tổng quát hóa và độ chính xác của mô hình.

Đánh giá: Mô hình đã được huấn luyện với sự kết hợp của hai bộ dữ liệu, giúp tăng dung lượng dữ liệu và tạo ra môi trường âm thanh đa dạng, phù hợp với thực tế, đặc biệt là trong ứng dụng **Telesales**. Tuy nhiên, số vòng lặp dài không mang lại nhiều hiệu quả, mà còn làm tăng khả năng **overfitting**. Trường hợp 3 là trường hợp ổn định nhất, với khả năng tổng quát tốt và đạt độ chính xác khoảng **82%**, nhưng vẫn cần tinh chỉnh thêm tham số để đạt được hiệu quả tối ưu.

### So sánh kết quả thực nghiệm mô hình Wav2Vec2 và Fast- Conformer

#### Bảng tổng kết kết quả huấn luyện mô hình Wav2Vec2 và Fast- Conformer.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Dataset | Tham số | | Kết quả | | | |
| Train\_loss | Val\_loss | Val\_wer | Test\_wer |
| 1 | Vivos | Lr Warmup\_steps  Lr\_scheduler\_type Epochs  Total steps | 1e-4  300  linear 20  1460 | 0.1664 | 0.315 | 0.2 | 0.1935 |
| 2 | Augmented Vivos | Lr Warmup\_ratio  Lr\_scheduler\_type Epochs  Total steps | 2e-4  0.2  linear 20  1460 | 0.1162 | 0.4755 | 0.2364 | 0.2251 |

**Bảng 9:** Kết quả huấn luyện tốt nhất các tập dữ liệu trên Wav2Vec2

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| STT | Dataset | Tham số | | Kết quả | | | | |
| Train\_loss | Val\_loss | Val\_wer | Test\_loss | Test\_we  r |
|  |  | Lr | 7e-4 |  |  |  |  |  |
|  |  | Warmup\_steps | 2000 |  |  |  |  |  |
| 1 | Tele | Min\_lr | 7e-5 | 0.7845 | 1.2236 | 0.1992 | 1.2528 | 0.2077 |
|  |  | Epochs | 20 |  |  |  |  |  |
|  |  | Total steps | 6000 |  |  |  |  |  |
|  |  | Lr | 2e-3 |  |  |  |  |  |
|  |  | Warmup\_ratio | 0.4 |  |  |  |  |  |
| 1 | Vivos | Min\_lr | 2e-4 | 0.46984 | 0.73309 | 0.14719 | 0.70878 | 0.1389 |
|  |  | Epochs | 20 |  |  |  |  |  |
|  |  | Total steps | 6000 |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | Augmented Vivos | Lr Warmup\_ratio Min\_lr Epochs  Total steps | 2e-3  0.3  2e-4  20  6000 | 0.76161 | 0.86282 | 0.20918 | 0.90096 | 0.20702 |
| 6 | Tele + Vivos | Lr Warmup\_ratio Min\_lr Epochs  Total steps | 3e-4  0.3  3e-5  20  11840 | 0.57483 | 1.03511 | 0.18337 | 1.0153 | 0.17753 |

**Bảng 10:** Kết quả huấn luyện tốt nhất các tập dữ liệu trên Fast-Conformer

#### Đánh giá hiệu suất suy luận mô hình Wav2Vec2 và Fast-Conformer.

Chúng tôi sử dụng tập dữ liệu nhỏ gồm 10 file âm thanh dạng .wav thu thập ngoài, có chất lượng âm thanh tương đương các file trên tập dữ liệu Augmented Vivos đã được tăng cường. Tương ứng với đó, mô hình Wav2Vec2 với Wer đạt 23,64% và Fast-Conformer với Wer đạt 20,92% trên tập dữ liệu Augmented Vivos để triển khai suy luận. Quá trình triển khai suy luận mô hình được đảm bảo trên cùng môi trường thực nghiệm.

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Model | Data | Tốc độ suy luận (s) |
| 1 | Wav2Vec2 - 23,64% - Augmented Vivos | 10 file audio thu thập ngoài, định dạng .wav | 0.8982 |
| 2 | Fast-Conformer - 20,92% - Augmented Vivos | 25.6288 |

**Bảng 11:** Kết quả suy luận triển khai Wav2Vec2 và Fast-Conformer

Với sự tối ưu trong quá trình tính toán, xử lý âm thanh, Fast-Conformer đã thể hiện khả năng suy luận với tốc độ vượt trội, so với mô hình Wav2Vec2.

#### Bảng so sánh kết quả huấn luyện mô hình Wav2Vec2 và Fast- Conformer

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **Wav2Vec2** | **Fast-Conformer** |
| **Kiến trúc mô hình** | Đơn giản, dựa trên Transformer. Chỉ sử dụng một kiến trúc Transformer đơn thuần để xử lý tín hiệu âm thanh và học các đặc trưng tự động từ dữ liệu thô. | Phức tạp hơn, kết hợp CNN và Transformer. Sử dụng các lớp CNN để nắm bắt các đặc trưng cục bộ, sau đó sử dụng Transformer để tổng quát hóa và học các mối quan hệ dài  hạn. |
| **Yêu cầu môi**  **trường** | Dễ dàng load model từ Hugging Face, có nhiều tài liệu và công cụ hỗ trợ giúp khởi  tạo và triển khai mô hình nhanh chóng. | Khó khăn hơn trong việc khởi tạo môi trường do yêu cầu nhiều hơn các thư viện và công cụ |

|  |  |  |
| --- | --- | --- |
| **thực**  **nghiệm** |  | hỗ trợ huấn luyện từ NVIDIA Nemo, làm cho  việc triển khai ban đầu phức tạp hơn. |
| **Yêu cầu dữ liệu** | Không cần tiền xử lý dữ liệu quá nhiều. Wav2Vec2 có thể học trực tiếp từ tín hiệu âm thanh thô mà không cần các bước chuyển đổi hoặc chuẩn bị phức tạp. | Âm thanh sẽ được chuyển thành Mel Spectrogram trước khi được đưa vào mô hình. Quá trình này giúp tăng cường dữ liệu thu thập được nhưng cũng đòi hỏi thêm bước  xử lý. |
| **Yêu cầu tài nguyên** | Cần nhiều tài nguyên tính toán, bao gồm GPU mạnh mẽ để xử lý lượng dữ liệu âm thanh lớn và huấn luyện các mô hình Transformer sâu. | Tối ưu hơn về tài nguyên, yêu cầu ít tài nguyên hơn so với Wav2Vec2 do cấu trúc phức hợp giữa CNN và Transformer, giúp tối ưu hóa hiệu suất mà không cần nhiều GPU  mạnh mẽ. |
| **Tốc độ huấn luyện, suy**  **luận** | Chậm. Do kiến trúc dựa hoàn toàn trên Transformer nên quá trình tính toán diễn ra chậm hơn, đặc biệt khi xử lý lượng lớn  dữ liệu âm thanh. | Nhanh hơn. Nhờ vào việc sử dụng CNN để xử lý các đặc trưng cục bộ, mô hình có thể suy luận nhanh hơn, đặc biệt khi kết hợp với  các công cụ tối ưu của Nemo. |
| **Khả năng nắm bắt ngữ cảnh** | Toàn cục. Mô hình nắm bắt toàn bộ chuỗi âm thanh đầu vào cùng một lúc, cho phép học các mối quan hệ dài hạn giữa các phần khác nhau trong tín hiệu âm thanh. Điều này giúp hiểu rõ hơn về  cấu trúc ngữ cảnh tổng thể. | Toàn cục + Cục bộ. Sự kết hợp giữa CNN và Transformer giúp mô hình không chỉ hiểu các mối quan hệ dài hạn mà còn nắm bắt được các đặc trưng cục bộ trong tín hiệu âm thanh, nhờ vậy có thể phản ứng nhanh với những  thay đổi nhỏ trong âm thanh. |
| **Hiệu suất mô hình** | Hoạt động tốt trên các tập dữ liệu sạch, âm thanh rõ ràng như Vivos. Tuy nhiên, gặp khó khăn khi triển khai trên dữ liệu có nhiều nhiễu hoặc đã được tăng cường, ví dụ như dữ liệu Tele hoặc Vivos có  nhiễu. | Dễ triển khai hơn trên các tập dữ liệu có nhiễu, có khả năng xử lý tốt cả những tập dữ liệu âm thanh khó và phức tạp. Nhờ sự kết hợp của CNN và Transformer, mô hình dễ dàng xử lý các thay đổi trong tín hiệu âm  thanh và vẫn giữ được hiệu suất tốt. |
| **Wer (Word Error**  **Rate)** | Vivos: 19,3%  Augmented Vivos: 22,5% | Vivos: 13,8%;  Augmented Vivos: 20,7%  Tele: 20,7%  Tele + Vivos: 17,7% |
| **Khả năng triển khai** | Phù hợp với môi trường dữ liệu sạch và không quá phức tạp. Dễ dàng triển khai trên các tập dữ liệu không nhiễu. | Linh hoạt hơn trong việc xử lý dữ liệu có nhiễu và dữ liệu đa dạng. Thích hợp cho các ứng dụng yêu cầu khả năng xử lý mạnh mẽ  với dữ liệu thực tế, nhiều nhiễu. |

**Bảng 12:** Bảng đánh giá quá trình thực nghiệm 2 mô hình Wav2Vec2 và Fast-Conformer

## Kết luận

Trong nghiên cứu này chúng tôi đã triển khai bài toán Nhận dạng giọng nói tự động sử dụng mô hình Wav2Vec2 và Fast-Conformer và đạt được những kết quả tích cực.

Wav2Vec2 đạt độ chính xác cao, tỉ lệ lỗi sai từ thấp, đạt Wer khoảng 19,3% trên tập dữ liệu sạch Vivos và 22,5% trên tập liệu Augmented Vivos được tăng cường. Fast-Conformer đạt Wer trung bình khoảng 20,7% trên tập dữ liệu Tele và Augmented Vivos được tăng cường, tốt nhất là 13,8% trên tập dữ liệu sạch Vivos và ổn định 17,7% trên tập dữ liệu kết hợp Tele và Vivos. Như vậy, Wav2Vec2 cho độ chính xác cao trên dữ liệu sạch (Vivos), trong khi Fast Conformer vượt trội hơn khi xử lý dữ liệu phức tạp, nhiễu (Tele), với tốc độ huấn luyện nhanh hơn và tiết kiệm tài nguyên. Chúng tôi đã tinh chỉnh các mô hình với bộ mã hóa Tokenizer, áp dụng các phương pháp tăng cường dữ liệu vừa phải, phù hợp, và tối ưu siêu tham số, giúp cải thiện hiệu suất tổng thể và khả năng triển khai suy luận trong thực.

Tuy nhiên, cả hai mô hình đều gặp phải một số thách thức chung khi xử lý các tập dữ liệu có thời lượng không đồng đều. Tập dữ liệu bài toán chưa đạt dung lượng lớn (trên 100h dữ liệu) khiến cả Wav2Vec2 và Fast-Conformer chưa thực sự đạt được kết quả huấn luyện tốt nhất.

Hướng phát triển tiếp theo:

* Tăng cường dung lượng bộ dữ liệu dùng cho bài toán ASR tiếng Việt, kết hợp nhiều nguồn dữ liệu hơn để phong phú, đa dạng hơn nữa ngữ cảnh dữ liệu, cải thiện tốc độ chính xác mô hình.
* Tích hợp mô hình ngôn ngữ (Language Model - LM) để cải thiện độ chính xác của đầu ra ASR. Mô hình ngôn ngữ có khả năng cung cấp ngữ cảnh ngữ nghĩa, giúp mô hình ASR giảm thiểu các lỗi phát sinh do nhận diện sai từ, từ đó nâng cao chất lượng và độ tin cậy của kết quả.
* Xây dựng các mô hình ASR đa ngôn ngữ, có khả năng xử lý giọng nói đa dạng, và cải thiện khả năng xử lý thời gian thực trên các thiết bị tài nguyên hạn chế. Tích hợp ASR với các hệ thống học máy khác, như dịch tự động hoặc tổng hợp giọng nói, sẽ mở ra cơ hội mới trong các lĩnh vực như chăm sóc sức khỏe, giáo dục, và dịch vụ khách hàng.

# PHỤ LỤC

Link tham khảo code Wav2Vec2 ASR: <https://github.com/ThienKym123/Wav2Vec2_ASR_Vivos.git> Link tham khảo code Fast-Conformer ASR:

<https://github.com/ThienKym123/Fast-Conformer.git>

# TÀI LIỆU THAM KHẢO

1. Research về Wav2vec2: “wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations”.

<https://arxiv.org/pdf/2006.11477>

1. Research về Conformer: “Conformer: Convolution-augmented Transformer for Speech Recognition”.

<https://arxiv.org/pdf/2005.08100>

1. How to using pretrain fast conformer ASR model for finetune task? <https://github.com/PhamDangNguyen/Fast-conformerASR-NVIDIA.git>