

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 đoán 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 đoán 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 bối cảnh thị trường tài chính ngày càng biến động, các tổ chức tài chính đang tích cực nghiên cứu và phát triển các hệ thống dự đoán thị trường chứng khoán– một chủ đề quan trọng trong lĩnh vực kinh tế và được quan tâm rộng rãi trong lĩnh vực khoa học dữ liệu. Việc xây dựng một hệ thống dự báo giá cổ phiếu hiệu quả không chỉ giúp nhà đầu tư đưa ra quyết định sáng suốt hơn, mà còn hỗ trợ các tổ chức tài chính trong việc quản trị rủi ro, tối ưu hóa danh mục đầu tư và phát hiện sớm các xu hướng thị trường tiềm năng. Tuy nhiên, bản chất phi tuyến tính, biến động mạnh và chịu ảnh hưởng bởi nhiều yếu tố kinh tế – xã hội khiến việc dự báo giá cổ phiếu trở thành một trong những bài toán phức tạp và đầy thách thức. Trong những năm gần đây, với sự phát triển mạnh mẽ của trí tuệ nhân tạo, đặc biệt là các mô hình học sâu (Deep Learning), độ chính xác và khả năng tổng quát trong dự đoán giá cổ phiếu đang dần được tăng lên.

Nghiên cứu này tập trung vào việc khai thác tiềm năng của mô hình học sâu trong việc xử lý và dự báo dữ liệu chuỗi thời gian tài chính, cụ thể là giá cổ phiếu của nhiều công ty khác nhau. Thay vì xây dựng mô hình dự báo riêng lẻ cho từng công ty, chúng tôi xây dựng một kiến trúc học sâu có tên là **CorrSTN (Correlation Information-based Spatiotemporal Network)** – một mô hình học sâu ban đầu được thiết kế cho bài toán **dự đoán lưu lượng giao thông** – để ứng dụng vào lĩnh vực tài chính. CorrSTN có khả năng học đồng thời mối quan hệ không gian (giữa các công ty) và quan hệ thời gian (trong chuỗi dữ liệu) để cải thiện hiệu quả dự báo. Việc chuyển đổi CorrSTN từ bài toán giao thông sang bài toán dự báo giá cổ phiếu không chỉ thể hiện tính linh hoạt của mô hình, mà còn mở ra hướng tiếp cận mới trong việc tận dụng mối quan hệ liên công ty để cải thiện hiệu quả dự đoán. Mục tiêu của công trình nghiên cứu này là nhằm kiểm chứng độ hiệu quả của mô hình CorrSTN trên dữ liệu thực tế, từ đó làm rõ tiềm năng ứng dụng của các kiến trúc học sâu trong lĩnh vực tài chính, đặc biệt trong bài toán dự báo giá cổ phiếu đa công ty dựa trên chuỗi thời gian.

## Mô hình CorrSTN

Mô hình CorrSTN (Correlation Information-based Spatiotemporal Network) là một mạng nơ-ron học sâu được thiết kế nhằm cải thiện hiệu quả dự đoán lưu lượng giao thông. Khác với các phương pháp truyền thống vốn chỉ sử dụng thông tin về không gian hoặc thời gian riêng lẻ, CorrSTN tận dụng triệt để thông tin tương quan giữa các chuỗi dữ liệu không gian – thời gian để xây dựng đặc trưng đầu vào chính xác và hiệu quả hơn.

CorrSTN bao gồm năm thành phần chính: SCorr (thông tin tương quan không gian), TCorr (thông tin tương quan thời gian), CIGNN (mạng nơ-ron đồ thị sử dụng thông tin tương quan), CIATT (attention với thông tin tương quan), và cuối cùng là kiến trúc Transformer với cơ chế encoder-decoder.

### SCorr – Thông tin tương quan không gian

SCorr là thành phần dùng để đo lường mức độ tương quan giữa các cảm biến giao thông trên nhiều thuộc tính, thông qua hệ số MIC (Maximal Information Coefficient). MIC có khả năng phát hiện cả các mối liên hệ tuyến tính lẫn phi tuyến giữa hai chuỗi dữ liệu. Nếu hai cảm biến có xu hướng lưu lượng thay đổi giống nhau theo thời gian, MIC giữa chúng sẽ cao, và SCorr cũng sẽ có giá trị lớn.

Khác với ma trận lân cận truyền thống chỉ cho biết hai cảm biến “có kết nối hay không” (giá trị 0 hoặc 1), SCorr biểu diễn mức độ tương quan bằng các giá trị liên tục từ 0 đến 1. Điều này cho phép mô hình hiểu rõ hơn về mức độ giống nhau về hành vi giữa các cảm biến, kể cả khi chúng không gần nhau về mặt địa lý.

### TCorr – Thông tin tương quan thời gian

TCorr được sử dụng để xác định mối liên hệ giữa dữ liệu hiện tại và dữ liệu trong các chu kỳ thời gian khác nhau như giờ, ngày hoặc tuần. Thay vì thử tất cả các tổ hợp dữ liệu theo kiểu truyền thống (exhaustive search), TCorr giúp mô hình tự động đánh giá dữ liệu nào (giờ, ngày, tuần) có ảnh hưởng nhiều nhất đến kết quả dự báo.

Nhờ vậy, CorrSTN có thể lựa chọn tập dữ liệu đầu vào hiệu quả hơn, giảm đáng kể chi phí tính toán và thời gian huấn luyện. TCorr không chỉ tăng tốc quá trình huấn luyện mà còn giúp mô hình tránh đưa vào những dữ liệu kém liên quan, từ đó cải thiện độ chính xác.

### CIGNN – Mạng nơ-ron đồ thị với thông tin tương quan

CIGNN là phiên bản cải tiến của mạng nơ-ron đồ thị (GNN), nơi việc lan truyền và tổng hợp thông tin giữa các nút (cảm biến) không chỉ dựa vào cấu trúc kết nối vật lý, mà còn được hướng dẫn bởi thông tin tương quan SCorr. Ngoài ra, CIGNN còn sử dụng ma trận trọng số động để thích ứng với sự thay đổi liên tục của dữ liệu trong quá trình huấn luyện.

Nhờ sự kết hợp giữa thông tin tương quan và cấu trúc đồ thị định sẵn, CIGNN có thể học được các đặc trưng chính xác và sâu sắc hơn, giúp tăng hiệu quả trích xuất đặc trưng không gian.

### CIATT – Attention với thông tin tương quan

CIATT là cơ chế attention được cải tiến từ attention truyền thống của Transformer. Thay vì tính toán trọng số chú ý giữa tất cả các vị trí trong chuỗi (có thể gây nhiễu), CIATT sử dụng SCorr để ưu tiên các chuỗi có mức độ tương đồng cao nhất với chuỗi hiện tại. Điều này giúp tập trung vào các phần tử có thông tin liên quan thực sự, đồng thời giảm ảnh hưởng của các phần tử không liên quan.

CIATT mang lại khả năng lọc nhiễu hiệu quả và làm tăng độ chính xác khi mô hình cố gắng tìm các mẫu tương tự trong chuỗi thời gian để dự báo tương lai.

### Kiến trúc Transformer với cơ chế Encode-Decoder

CorrSTN được triển khai dựa trên kiến trúc Transformer dạng encoder–decoder. Đây là kiến trúc mạnh mẽ trong xử lý chuỗi, nổi bật nhờ khả năng học được các quan hệ dài hạn và phi tuyến giữa các phần tử trong chuỗi. Encoder tiếp nhận dữ liệu lịch sử (theo chu kỳ giờ, ngày, tuần) và mã hóa thành đặc trưng trừu tượng; decoder sử dụng các đặc trưng này để tạo ra chuỗi dự báo theo từng bước.

Điểm khác biệt quan trọng trong CorrSTN so với các mô hình Transformer truyền thống là việc tích hợp CIGNN và CIATT vào từng lớp encoder và decoder. Nhờ đó, mỗi bước xử lý của mô hình không chỉ học từ chuỗi thời gian mà còn đồng thời cập nhật các mối quan hệ giữa các cảm biến và giữa các chu kỳ. Sự kết hợp này giúp CorrSTN xử lý dữ liệu giao thông có cấu trúc phức tạp một cách hiệu quả và chính xác hơn, tạo ra dự báo ổn định và tin cậy hơn trong các hệ thống giao thông thông minh.

1. **Áp dụng mô hình CorrSTN trong dự đoán chứng khoán**

Mặc dù mô hình CorrSTN ban đầu được phát triển để dự đoán lưu lượng giao thông, nhưng mô hình này có thể dễ dàng áp dụng cho các bài toán dự đoán giá chứng khoán, nhờ vào những đặc điểm chung giữa hai vấn đề này. Cả dự đoán giao thông và chứng khoán đều liên quan đến việc xử lý chuỗi thời gian, trong đó các yếu tố không gian và thời gian đóng vai trò quan trọng trong việc dự đoán.

Mô hình CorrSTN đặc biệt hiệu quả trong việc khai thác mối quan hệ không gian và thời gian giữa các yếu tố dữ liệu. Trong trường hợp giao thông, các cảm biến giao thông trên các tuyến đường có thể có sự tương tác lẫn nhau, và trong chứng khoán, các cổ phiếu của các công ty có thể có mối quan hệ tương tự, đặc biệt khi chúng thuộc cùng một ngành hoặc cùng chịu ảnh hưởng từ các yếu tố kinh tế vĩ mô. Mô hình sử dụng SCorr (Spatial Correlation Information) để phát hiện các mối quan hệ không gian giữa các cảm biến và TCorr (Temporal Correlation Information) để nhận diện các mẫu biến động theo chu kỳ trong dữ liệu, giúp tối ưu hóa dự báo.

Với khả năng khai thác thông tin từ dữ liệu chu kỳ (như theo giờ, ngày, tuần), CorrSTN có thể dễ dàng áp dụng cho dữ liệu chứng khoán, nơi các dữ liệu theo chu kỳ (ngày, tuần, tháng) cũng có ảnh hưởng lớn đến sự biến động của giá cổ phiếu. Thêm vào đó, CIGNN (Correlation Information Graph Neural Network) và CIATT (Correlation Information Multi-Head Attention) trong mô hình giúp khai thác các mối quan hệ phức tạp giữa các yếu tố không gian và thời gian, giúp dự đoán chính xác hơn các biến động của giá chứng khoán.

Vì thế, mô hình CorrSTN không chỉ có thể áp dụng trong các hệ thống giao thông thông minh mà còn có tiềm năng lớn trong việc dự đoán giá chứng khoán, đặc biệt là trong việc phân tích các mối quan hệ giữa các cổ phiếu và nhận diện các chu kỳ thay đổi của thị trường tài chính.

## Thực nghiệm

### Bài toán thực nghiệm

Mục tiêu của thực nghiệm là đánh giá khả năng áp dụng mô hình CorrSTN vào bài toán dự đoán giá cổ phiếu, thông qua việc sử dụng dữ liệu thực tế từ thị trường chứng khoán. Thực nghiệm được thiết kế nhằm kiểm tra xem mô hình có thể học được các đặc trưng không gian – thời gian giữa các cổ phiếu và dự đoán chính xác giá trị tương lai của chúng hay không. Việc này không chỉ giúp xác minh tính khả chuyển của CorrSTN từ lĩnh vực giao thông sang tài chính, mà còn cung cấp một góc nhìn mới trong việc ứng dụng các mô hình học sâu tiên tiến vào phân tích và dự đoán thị trường.

Dữ liệu đầu vào bao gồm chuỗi thời gian giá cổ phiếu của 10 công ty được thu thập thông qua Yahoo Finance API. Các công ty được lựa chọn thuộc nhiều ngành nghề khác nhau, nhằm đảm bảo tính đa dạng và phản ánh rõ các mối quan hệ không gian giữa các cổ phiếu. Mỗi công ty bao gồm các đặc trưng như: giá mở cửa (Open), giá đóng cửa (Close), giá cao nhất (High), giá thấp nhất (Low), khối lượng giao dịch (Volume), và được thu thập theo chu kỳ ngày. Dữ liệu được chia thành tập huấn luyện, kiểm tra và kiểm định để đánh giá hiệu suất của mô hình một cách khách quan.

Dữ liệu đầu ra là giá cổ phiếu của từng công ty trong một hoặc nhiều bước thời gian tương lai, được dự đoán dựa trên dữ liệu lịch sử và mối tương quan với các cổ phiếu khác trong danh mục. Trong quá trình dự đoán, mô hình tận dụng thông tin tương quan về hành vi giá giữa các cổ phiếu cũng như xu hướng biến động theo thời gian, tương tự như cách CorrSTN xử lý mối quan hệ giữa các cảm biến trong bài toán giao thông.

Thực nghiệm được tiến hành nhằm kiểm tra độ chính xác, khả năng học đặc trưng và mức độ ổn định của mô hình CorrSTN trong bối cảnh mới – thị trường tài chính – từ đó làm nền tảng cho việc đánh giá tính ứng dụng rộng rãi của mô hình trong các bài toán chuỗi thời gian đa chiều có yếu tố tương quan cao.

### Môi trường thực nghiệm

Toàn bộ quá trình thực nghiệm được thực hiện trên nền tảng Google Colab, sử dụng ngôn ngữ lập trình Python cùng các thư viện như NumPy, Scikit-learn và PyTorch để xây dựng và huấn luyện mô hình. Google Colab được lựa chọn nhờ khả năng hỗ trợ GPU miễn phí, dễ dàng truy cập và tích hợp với Google Drive. Dữ liệu chứng khoán được thu thập thông qua Yahoo Finance API, lưu trữ trực tiếp trên Drive và xử lý ngay trong môi trường Colab, giúp tối ưu hiệu quả thử nghiệm và thuận tiện cho việc tái sử dụng, mở rộng nghiên cứu.

### Dữ liệu thực nghiệm

Dữ liệu được sử dụng trong thực nghiệm là chuỗi thời gian giá cổ phiếu của 10 công ty niêm yết trên thị trường chứng khoán Mỹ, được thu thập thông qua Yahoo Finance API. Các công ty được lựa chọn thuộc nhiều lĩnh vực khác nhau như công nghệ, tài chính, năng lượng và y tế nhằm đảm bảo tính đa dạng và cho phép mô hình khai thác các mối quan hệ tương quan giữa các ngành nghề trong thị trường. Dữ liệu được lấy theo chu kỳ ngày (daily) trong khoảng thời gian từ 1/1/2018 đến 31/12/2023, bao gồm các đặc trưng phổ biến như: giá mở cửa (Open), giá cao nhất (High), giá thấp nhất (Low), giá đóng cửa (Close), và khối lượng giao dịch (Volume).

### Thu thập và tiền xử lý dữ liệu

Dữ liệu được thu thập từ **Yahoo Finance API**, sau đó được chuẩn hóa bằng phương pháp **Min-Max normalization** để đưa tất cả giá trị về khoảng [−1,1]. Điều này giúp đồng nhất các đặc trưng, đồng thời loại bỏ các giá trị thiếu và giảm thiểu sự ảnh hưởng của các biến động lớn trong dữ liệu. Sau khi tiền xử lý, dữ liệu được chia thành các tập huấn luyện, kiểm tra và kiểm định. Việc chia dữ liệu được thực hiện theo **trình tự thời gian,** từ các mẫu đầu chuỗi cho đến cuối chuỗi, nhằm đảm bảo mô hình học được các quy luật theo thời gian mà không bị rò rỉ thông tin từ tương lai.

### Mã hóa dữ liệu đầu vào

Sau khi dữ liệu được chuẩn hóa và chia theo trình tự thời gian, mỗi mẫu dữ liệu được tổ chức dưới dạng tensor ba chiều với cấu trúc: (số lượng mẫu, số lượng cổ phiếu, số bước thời gian). Trước khi đưa vào mô hình, dữ liệu này được biến đổi qua một lớp tuyến tính nhằm đưa các đặc trưng về không gian đặc trưng có số chiều phù hợp với kiến trúc mạng nơ-ron.

Tiếp theo, dữ liệu được mã hóa thông qua các lớp encoder của mô hình CorrSTN. Mỗi lớp encoder kết hợp giữa cơ chế attention và mạng nơ-ron đồ thị (GCN). Attention giúp mô hình học được các mối liên hệ theo thời gian, trong khi GCN khai thác mối quan hệ không gian giữa các cổ phiếu thông qua ma trận tương quan. Quá trình này cho phép mô hình trích xuất ra các đặc trưng quan trọng, thể hiện được xu hướng biến động của giá cổ phiếu và mối tương quan giữa các công ty trong danh mục đầu tư. Đây là bước quan trọng để chuẩn bị cho quá trình giải mã và dự đoán ở các bước tiếp theo.

### Xây dựng mô hình

Mô hình CorrSTN được xây dựng thông qua một hàm khởi tạo mô hình bao gồm đầy đủ các thành phần từ mã hóa đến dự đoán. Quá trình xây dựng mô hình bắt đầu bằng việc tạo ra các ma trận tương quan không gian, bao gồm ma trận lân cận được chuẩn hóa từ dữ liệu cấu trúc đồ thị và ma trận SCorr từ hệ số tương quan MIC. Hai ma trận này đóng vai trò quan trọng trong việc truyền thông tin giữa các cổ phiếu trong quá trình học không gian.

Tiếp theo, các lớp embedding được khởi tạo để mã hóa dữ liệu đầu vào và đầu ra. Quá trình mã hóa bao gồm hai phần: mã hóa vị trí thời gian (temporal positional encoding) để mô hình nhận biết vị trí tương đối của dữ liệu trong chuỗi thời gian, và mã hóa không gian (spatial positional encoding) để thể hiện đặc trưng của từng cổ phiếu trong mạng lưới. Các mã hóa này được áp dụng lên dữ liệu đầu vào thông qua các lớp tuyến tính.

CorrSTN sử dụng các cơ chế attention khác nhau tùy thuộc vào cấu hình, bao gồm attention truyền thống và attention có nhận thức về ngữ cảnh thời gian (temporal context-aware). Ba loại attention chính được sử dụng trong mô hình là: attention giữa các bước thời gian trong encoder, attention giữa dữ liệu đầu ra và dữ liệu đã mã hóa, và attention trong decoder theo hướng nhân quả (causal), tức là mỗi bước thời gian chỉ được phép truy cập các thông tin từ quá khứ đến hiện tại, giúp đảm bảo tính hợp lý và khả năng ứng dụng của mô hình trong thực tế..

Sau khi mã hóa, các lớp encoder và decoder được xây dựng từ nhiều tầng (layers), mỗi tầng bao gồm: attention, mạng nơ-ron đồ thị (GCN), kết nối dư (residual connection) và chuẩn hóa lớp (layer normalization). Các tầng này cho phép mô hình học được các mối quan hệ phức tạp về không gian và thời gian trong dữ liệu đầu vào.

Cuối cùng, đầu ra của decoder được đưa qua một lớp tuyến tính để chuyển về đúng số chiều đầu ra cần dự đoán (ví dụ: giá cổ phiếu trong tương lai). Tất cả các tham số trong mô hình được khởi tạo bằng phương pháp Xavier initialization để đảm bảo mô hình học hiệu quả và ổn định ngay từ đầu.

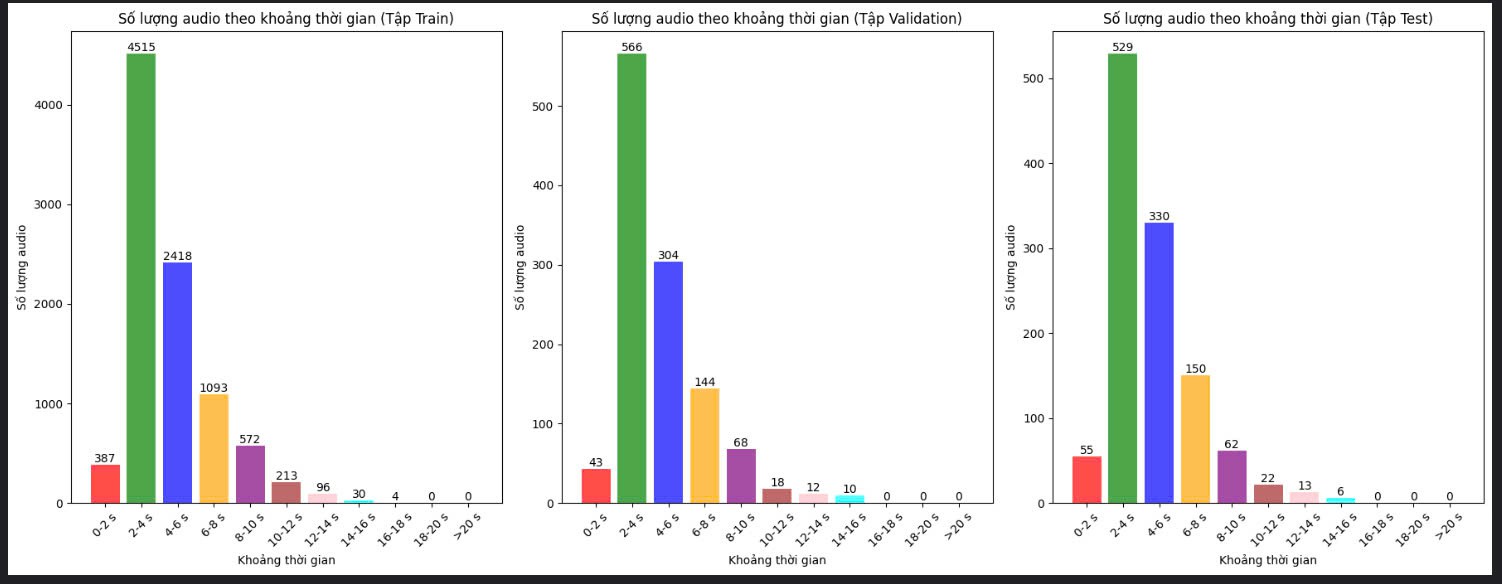
## Kết quả thực nghiệm

### 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



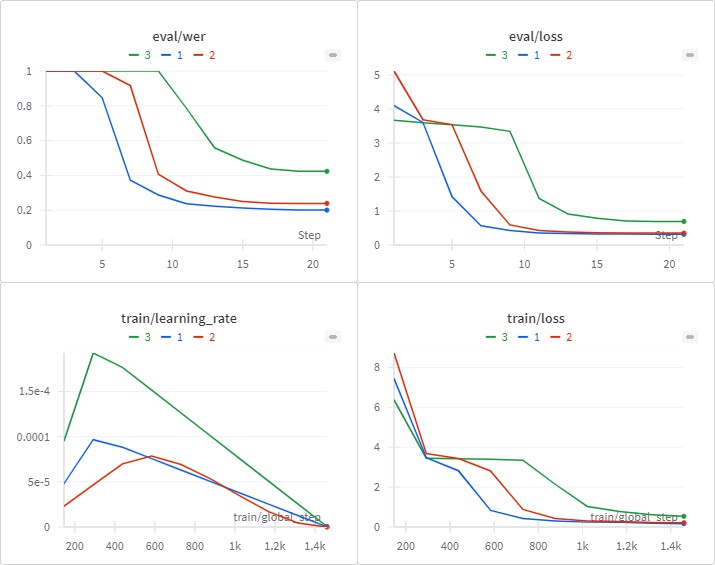
**Hình 6:** Thống kê số lượng bản ghi Vivos theo thời lượng Bảng kết quả các lần thử nghiệm

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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

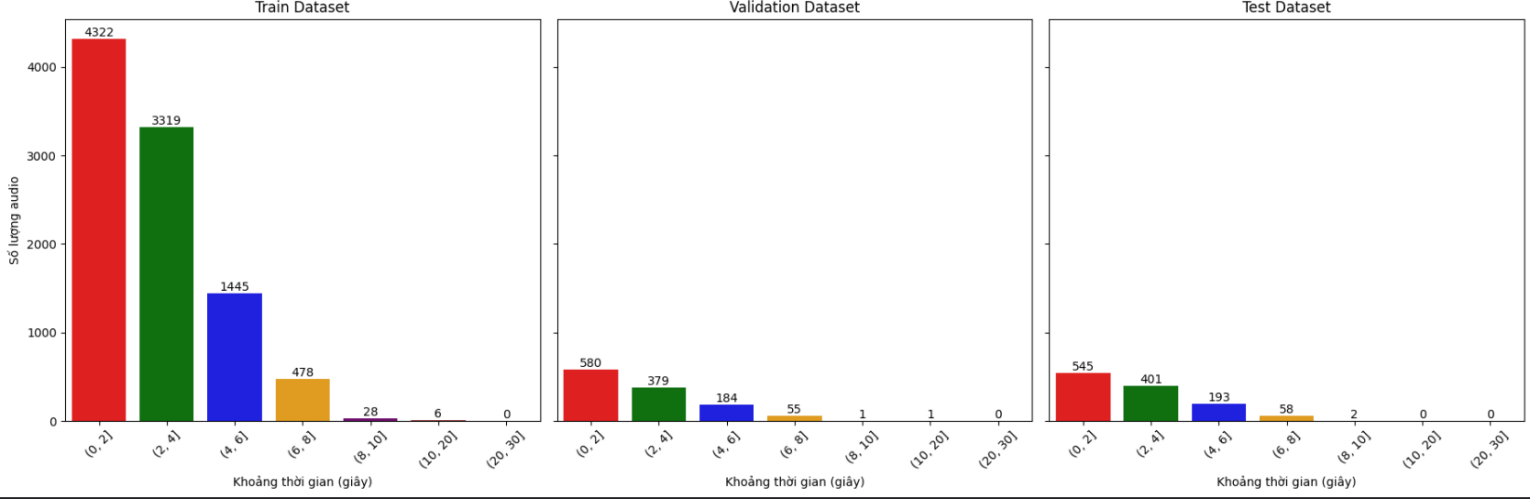
Triển khai bài toán ASR trên các dataset: **Tele**, **Vivos**, **Tele + Vivos**.

#### 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**.

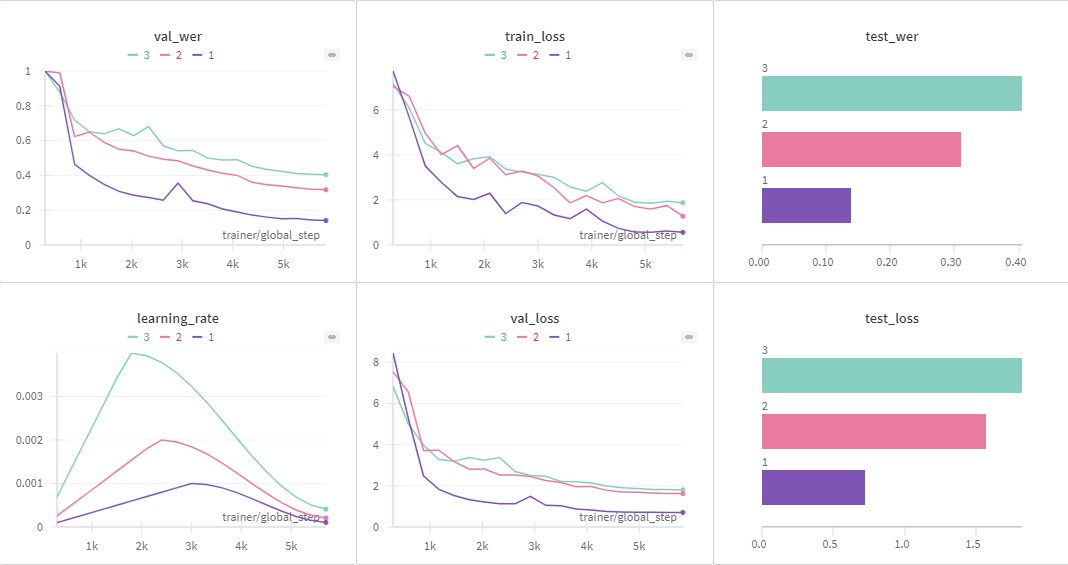
Đánh giá: Mô hình đạt sự ổn định tốt nhất trên tập dữ liệu **Tele** với ngữ cảnh trong lĩnh vực **Telesales** ở trường hợp 2, nơi **learning rate** hợp lý giúp đạt độ chính xác tốt mà không gây quá nhiều **overfitting**. Việc huấn luyện mô hình qua nhiều vòng lặp không cải thiện đáng kể độ chính xác 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.

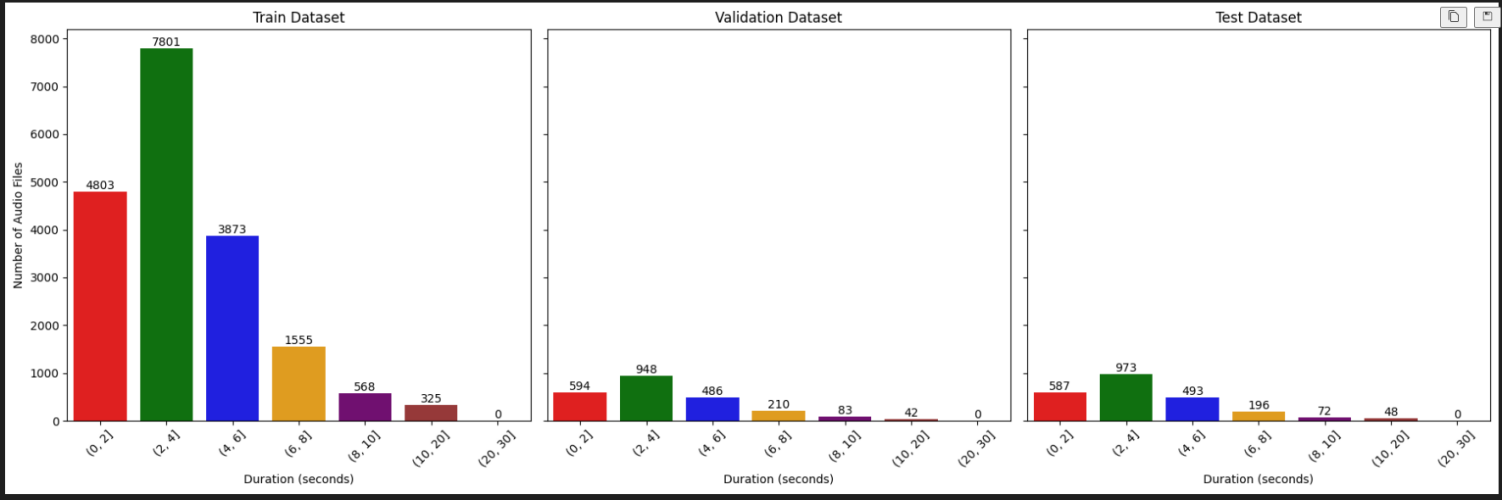
Đánh giá: Mô hình cho thấy khả năng tổng quát hóa tốt khi làm việc với tập dữ liệu có ngữ cảnh đa dạng và rõ ràng, không có nhiễu, với độ chính xác khoảng 86%. Tuy nhiên, khi làm việc với dữ liệu đã được tăng cường và có nhiễu, mô hình vẫn có thể xử lý, nhưng cần điều chỉnh tỉ lệ nhiễu một cách phù hợp 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 |

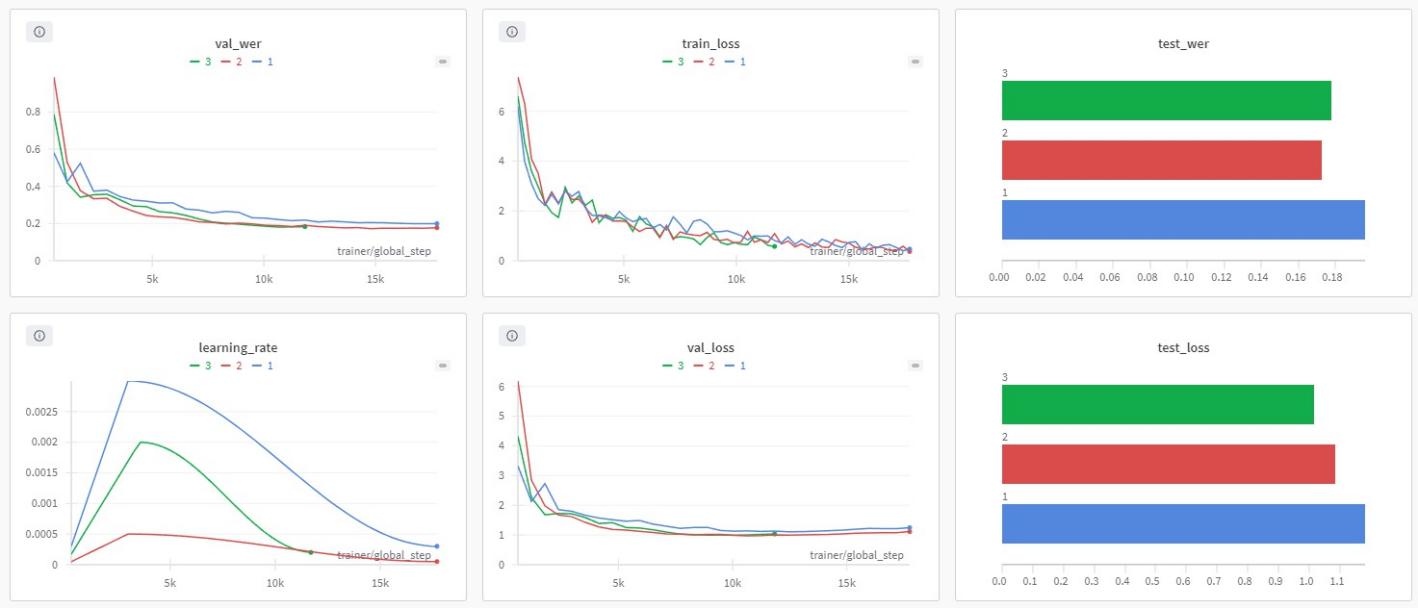
**Bảng 7:** Dung lượng tập dữ liệu Tele + Vivos



**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>