[CHƯƠNG 1. 3](#_Toc59616827)

[CHƯƠNG 2. 3](#_Toc59616828)

[CHƯƠNG 3. GIẢI PHÁP XÂY DỰNG MÔ HÌNH ÂM THANH 3](#_Toc59616829)

[3.1. Tổng quan kiến trúc hệ thống 3](#_Toc59616830)

[3.2. Giải pháp tiền xử lí âm thanh 5](#_Toc59616831)

[3.2.1. Tổng quan giải pháp 5](#_Toc59616832)

[3.2.2. Chi tiết giải pháp 6](#_Toc59616833)

[3.3. Giải pháp xây dựng mô hình âm thanh 10](#_Toc59616834)

[3.3.1. Tổng quan giải pháp 10](#_Toc59616835)

[3.3.2. Mô hình mạng nơ-ron hồi quy RNN 10](#_Toc59616836)

[3.3.3. Chuẩn hóa hàng loạt (Batch normalization) 15](#_Toc59616837)

[3.3.4. SortaGrad 16](#_Toc59616838)

[3.3.5. Biến đổi tần số (Freequency Convolution) 16](#_Toc59616839)

[3.3.6. Bước sóng (Striding) 17](#_Toc59616840)

[3.4. Giải pháp xây dựng bộ giải mã 17](#_Toc59616841)

[3.4.1. Bộ giải mã tham lam 17](#_Toc59616842)

[3.4.2. Bộ giải mã tìm kiếm chùm với mô hình ngôn ngữ 19](#_Toc59616843)

[3.4.3. Mô hình ngôn ngữ (Language model) 20](#_Toc59616844)

[3.5. Giải pháp xây dựng máy chủ 22](#_Toc59616845)

[3.6. Giải pháp xây dựng ứng dụng mẫu 23](#_Toc59616846)

[CHƯƠNG 4. QUÁ TRÌNH HUẤN LUYỆN MÔ HÌNH 25](#_Toc59616847)

[4.1. Mô hình huấn luyện được sử dụng 25](#_Toc59616848)

[4.2. dữ liệu huấn luyện MÔ HÌNH 25](#_Toc59616849)

[4.3. Xây dựng language model 27](#_Toc59616850)

[4.4. Các lỗi huấn luyện 27](#_Toc59616851)

[4.4.1. Mô hình DeepSpeech 2 27](#_Toc59616852)

[4.5. Kết quả huấn luyện 32](#_Toc59616853)

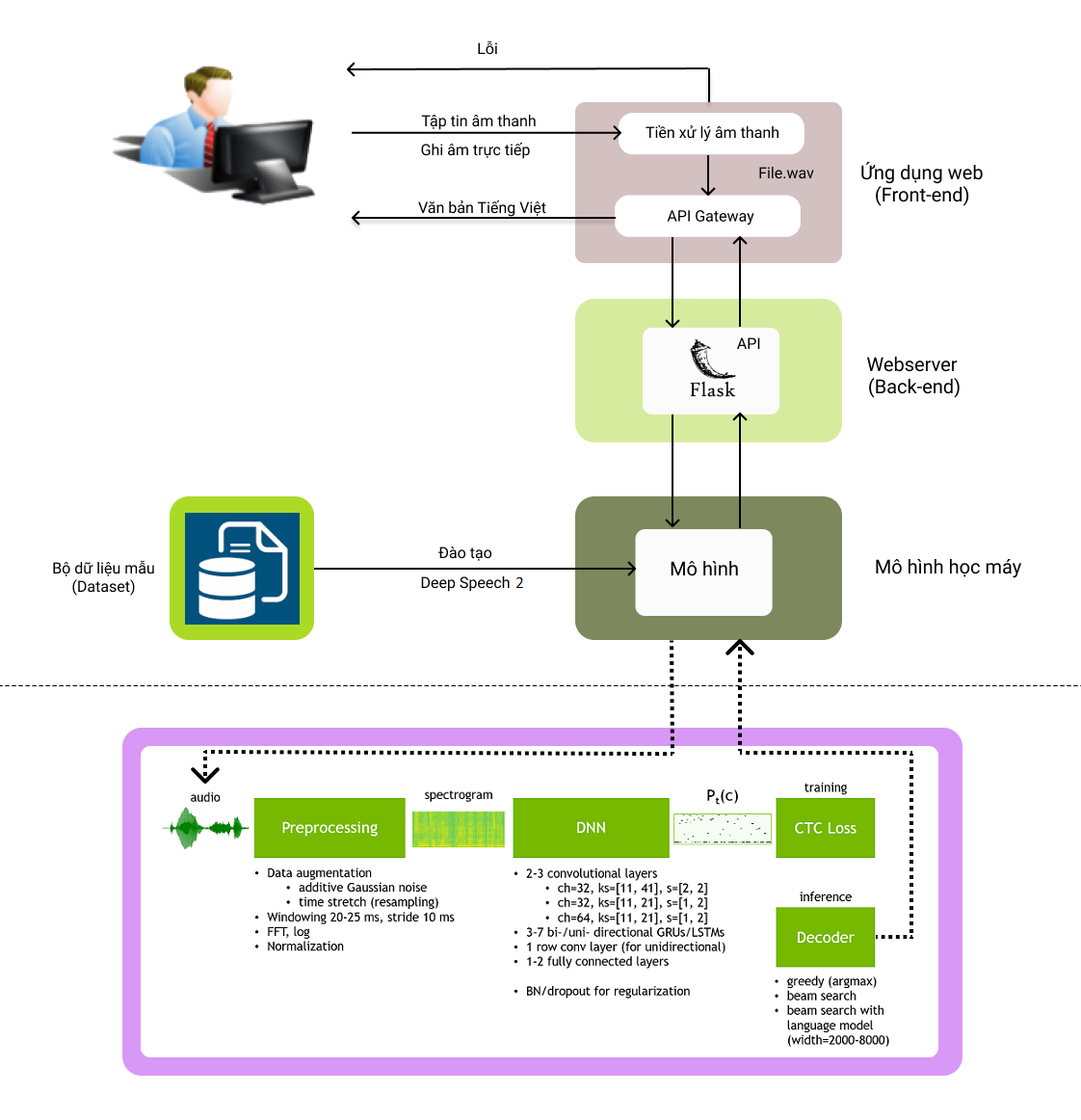
# 

# 

# GIẢI PHÁP XÂY DỰNG MÔ HÌNH ÂM THANH

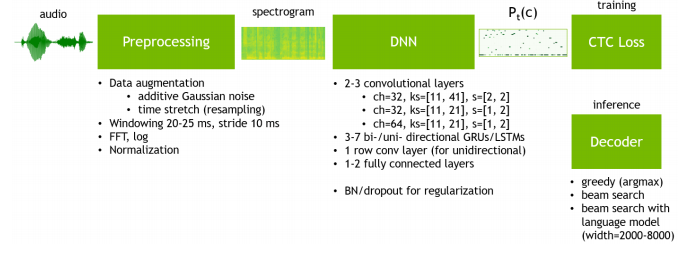
## Tổng quan kiến trúc hệ thống

Nhóm sinh viên dự kiến xây dựng sản phẩm là trang web với kiến trúc và mô hình được minh họa trong hình 3.1.



Hình 3.1

Đề tài dự kiến sử dụng kiến trúc mô hình nhận dạng đầu cuối (End-to-End), được xây dựng kết hợp cùng ý tưởng DeepSpeech 2, một nghiên cứu của Baidu được công bố vào ngày 08/12/2015 tại Silicon Valley AI Lab. Nội dung của bài báo trình bày về nhận dạng giọng nói được thực hiện trên ngôn ngữ Tiếng Anh (English) và tiếng Quan Thoại (Mandarin).

Deep Speech 2 sử dụng mô hình Mạng nơ-ron hồi quy (RNN – Recurrent Neural Network) và sử dụng Connectionist Temporal Classification (CTC) để dự đoán đầu ra:

Đầu vào của hệ thống là đoạn âm thanh thô chưa được xử lý. Phần tiền xử lý (Preprocessing) lấy một tín hiệu dạng sóng âm thanh thô và chuyển nó thành một biểu đồ phổ có kích thước (N\_timesteps, N\_frequency\_features). N\_timesteps phụ thuộc vào thời lượng của tệp âm thanh gốc, N\_frequency\_features có thể được chỉ định trong tệp cấu hình của mô hình dưới dạng thông số “num\_audio\_features”.

Phần Deep Neural Network (DNN) tạo ra phân phối xác suất Pt(c) trên các ký tự từ vựng c cho mỗi bước thời gian *t*. Deep Speech 2 được đào tạo với nhiều thử nghiệm với mạng nơ-ron được huấn luyện với chức năng suy giảm phân loại theo thời gian kết nối (CTC) để dự đoán phiên âm giọng nói từ âm thanh. Tỷ lệ lỗi từ (WER) là số liệu đánh giá độ chính xác của mô hình.

Để đưa các từ ra khỏi một mô hình được đào tạo, cần sử dụng một bộ giải mã (Decoder). Bộ giải mã chuyển đổi phân phối xác suất trên các ký tự thành văn bản. Có hai loại bộ giải mã thường được sử dụng với các mô hình dựa trên CTC: bộ giải mã tham lam (Greedy decoder) và bộ giải mã tìm kiếm chùm (Beam search decoder) với mô hình ngôn ngữ. Một bộ giải mã tham lam xuất ra ký tự có thể xảy ra nhất ở mỗi bước thời gian. Nó có tốc độ xử lý nhanh và có thể tạo ra các câu rất chính xác, nhưng có thể mắc nhiều lỗi chính tả nhỏ. Tuy nhiên, do bản chất của chỉ số WER, một lỗi ký tự cũng làm cho một từ không chính xác. Một bộ giải mã tìm kiếm chùm có chức năng ghi lại mô hình ngôn ngữ cho phép kiểm tra nhiều giải mã bằng cách chỉ định điểm cao hơn cho nhiều N-grams tùy vào mô hình ngôn ngữ nhất định. Mô hình ngôn ngữ cũng giúp sửa lỗi chính tả. Nhược điểm là nó chậm hơn đáng kể so với một bộ giải mã tham lam.

Đầu ra của hệ thống là một đoạn văn bản Tiếng Việt hoàn chỉnh.

## Giải pháp tiền xử lí âm thanh

### Tổng quan giải pháp

Để xử lý dữ liệu, âm thanh dạng sóng chuyển đổi thành chương trình quang phổ và cấp cho mạng nơ-ron để tạo ra đầu ra. Cách truyền thống để thực hiện tăng dữ liệu thường được áp dụng cho dạng sóng. Bên cạnh đó còn một cách tiếp cận khác đó là thao tác trên phổ.

Nhóm sinh viên đề xuất sử dụng kỹ thuật Short Time Fourier Transform (STFT) với mục đích tiếp cận âm thanh của tiếng nói tiếng Việt dưới dạng phổ, cụ thể hơn là một chuỗi các vector n chiều (mỗi chiều là một giá trị thực) để phục vụ cho quá trình huấn luyện.

Biến đổi Fourier thời gian ngắn (STFT) là một chuỗi các biến đổi Fourier của tín hiệu cửa sổ. STFT cung cấp thông tin tần số được chuẩn hóa theo thời gian cho các tình huống trong đó các thành phần tần số của tín hiệu thay đổi theo thời gian, trong khi biến đổi Fourier tiêu chuẩn cung cấp thông tin tần số được tính trung bình trong toàn bộ khoảng thời gian của tín hiệu.

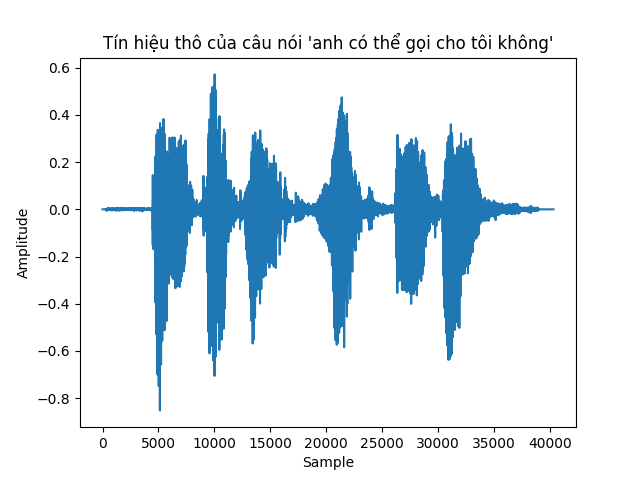
Về cơ bản phép biến đổi Fourier là quá trình phân hủy âm thanh tuần hoàn thành tổng các sóng sin mà tất cả các sóng dao động đều dao động ở các tần số khác nhau. Thật là khó tin vì vậy chúng ta có thể mô tả một âm thanh rất phức tạp miễn là nó có tính chu kỳ là một tổng như sự chồng chất của một loạt các sóng sin khác nhau ở các tần số khác nhau.

### Chi tiết giải pháp

Nhóm sinh viên sử dụng đầu vào là tập tin WAV PCM 16 bit, với tần số lấy mẫu 16000 Hz, thời lượng không vượt quá 10s. Tập tin WAV là tín hiệu giọng nói Tiếng Việt rõ ràng (hoặc bị nhiễu).

Minh họa ví dụ dưới đây nhóm sinh viên sử dụng câu nói: “anh có thể gọi tôi không”

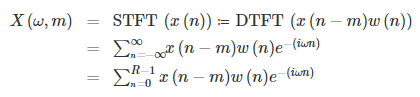
Tín hiệu thô sẽ được biểu diễn với âm độ như sau trong miền thời gian:



Hình 1.1 Tín hiệu thô trong miền thời gian của câu nói “anh có thể gọi tôi không”

Bước đầu tiên là áp dụng phân tích tín hiệu thô thành phổ sử dụng STFT. Phép biến đổi Fourier cổ điển được thiết kế để chuyển từ biểu diễn thời gian của tín hiệu sang biểu diễn tần số và ngược lại. Tuy nhiên, tín hiệu lời nói thường có đặc tính là các đặc tính tần số của chúng thay đổi tương đối chậm theo thời gian. Do đó, sẽ rất hữu ích nếu sử dụng một phép biến đổi hiển thị nội dung phổ của tín hiệu giọng nói như một hàm của thời gian. Biến đổi Fourier trong thời gian ngắn là một biến đổi như vậy. STFT là một phép biến đổi từ miền thời gian sang miền tần số thời gian. Nó tạo ra một biểu diễn "quang phổ" hai chiều. Thực tế là biểu diễn này có chứa trục tần số cũng như trục thời gian, một mặt mang lại sự hiểu biết còn thiếu trong trường hợp biểu diễn tần số hoặc thời gian thuần túy, mặt khác nó tạo ra một số vấn đề phụ.

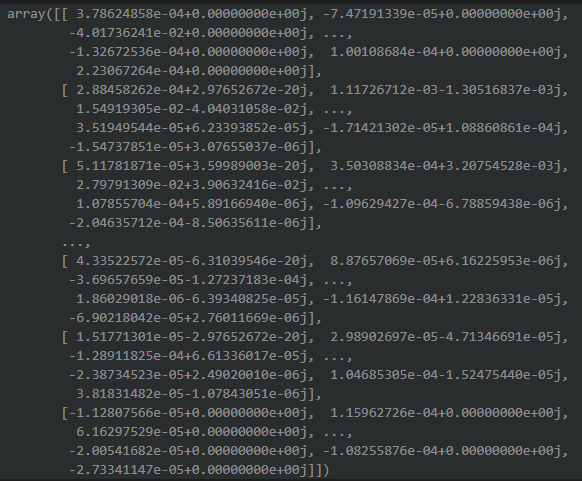
Biến đổi Fourier F (w) của một tín hiệu thực / (t) thường sẽ bao gồm các số phức. Bộ lọc tiền nhấn mạnh có thể được áp dụng vào tín hiệu x với công thức sau:



trong đó:

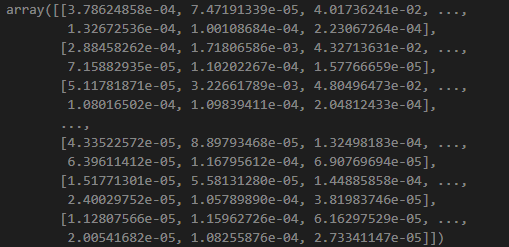
* w (n) là hàm cửa sổ có độ dài RR.
* STFT của tín hiệu x (n) là một hàm của hai biến: thời gian và tần số.

Ma trận số phức của phép biến đổi STFT của câu nói “anh có thể gọi tôi không” với thông số chiều dài của cửa sổ phân tích (winlen) là 320 và số lượng mẫu âm thanh giữa các cột STFT liền kề (hop\_length) là 160, và độ dài của tín hiệu được cửa sổ sau khi đệm bằng các số không (n\_fft) là 320, sau đó áp dụng hàm phân tích Hamming lên.

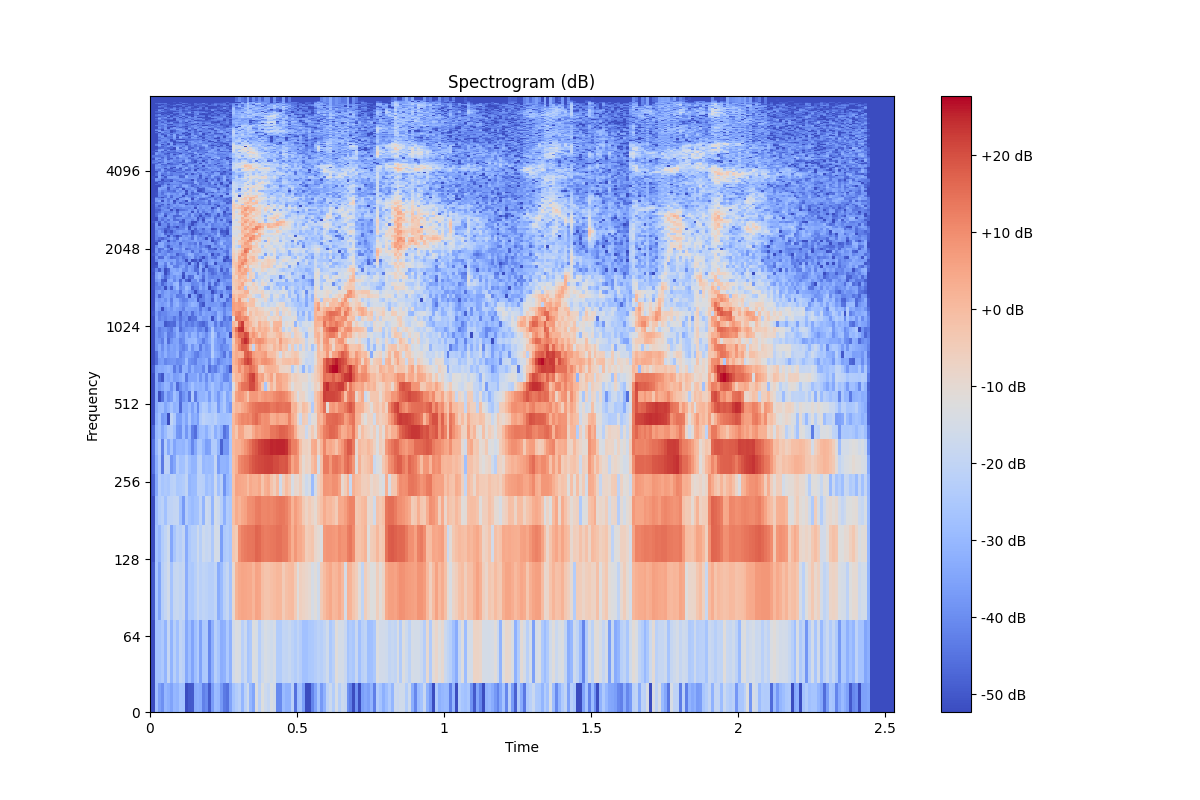


Hình 1.2. Biến đổi STFT của câu nói “anh có thể gọi cho tôi không”,ma trận số phức

Bước tiếp theo, tách một phổ có ma trận số phức D thành các thành phần độ lớn (S) và pha (P) của nó, khi đó phổ là một ma trận số thực như sau:



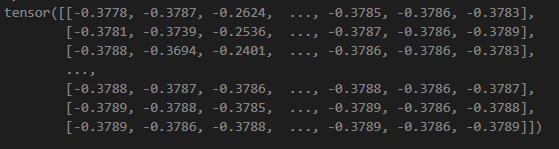
Mô tả phổ của ma trận số thực đó như sau:



Hình 1.3. Phổ của câu nói “anh có thể gọi tôi không”

Bước cuối cùng phổ đầu vào được chuẩn hóa hàng loạt (Batch Normailization), sẽ được nhóm trình bày trong phần sau của chương, ta có được ma trận phổ đầu vào cuối cùng là một tensor các số thực.

Kết quả tiền xử lí âm thanh vào được biểu diễn ở hình :



Hình 1.4. Ma trận sau khi tiền xử lí âm thanh của câu nói “anh có thể gọi tôi không”

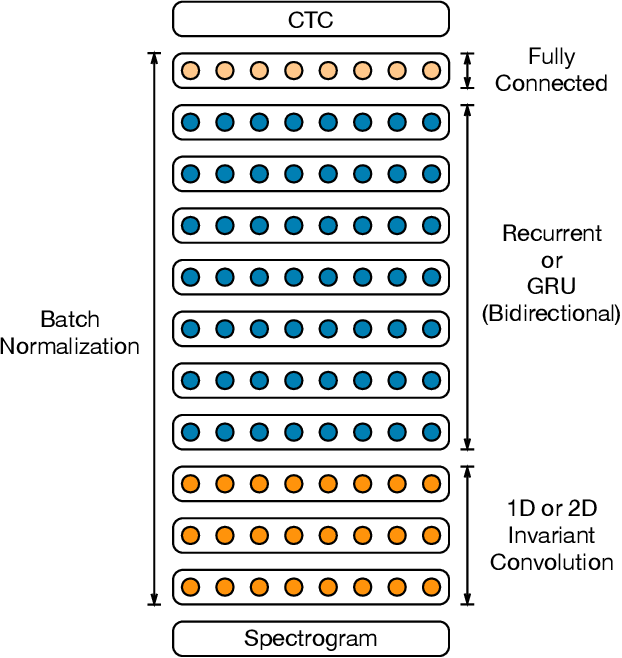
## Giải pháp xây dựng mô hình âm thanh

### Tổng quan giải pháp

Một mô hình nhiều lớp đơn giản với một lớp hồi quy không thể khai thác hàng nghìn giờ dữ liệu lời nói đã được gắn nhãn. Để mô hình học tập được tập dữ liệu lớn, cần phải tăng dung lượng mô hình thông qua chiều sâu. Nhóm sinh viên dự kiến sử dụng kiến trúc mô hình có 11 lớp bao gồm nhiều lớp hồi quy hai chiều (bidirectional recurrent) và lớp tích tụ (convolutional). Đồng thời, để tối ưu hóa mô hình, nhóm sử dụng chuẩn hóa hàng loạt cho RNN (Batch Normalization) và một chương trình tối ưu hóa mới gọi là SortaGrad. Đồng thời nhóm cũng khai thác những bước tiến dài (long strides) giữa các đầu vào RNN để giảm việc tính toán cho mỗi ví dụ theo hệ số 3.

### Mô hình mạng nơ-ron hồi quy RNN

Một mạng nơ tron hồi quy được đào tạo để nhận các âm phổ đầu vào và cho ra các ma trận thể hiện xác suất suất hiện của một kí tự trong bảng kí tự Tiếng Việt, sau đó sử dụng bộ giải mã để đạt được văn bản Tiếng Việt cuối cùng. Kiến trúc mô hình RNN được minh họa trong hình.



Hình 1.5. Kiến trúc hệ thống DeepSpeech2

Cụ thể, trong một tập huấn luyện với x là phát âm đầu vào và y là nhãn văn bản Tiếng Việt tương ứng với đầu ra. Mỗi phát âm là một chuỗi thời gian có độ dài là . Trong đó, mỗi lát cắt thời gian nhất định của là một vector của các đặc trưng âm thanh , t = 1 ,2 , ... .

Nhóm sinh viên tiếp tục sử dụng âm phổ của các đoạn âm thanh được chuẩn hóa bằng STFT được trình bày ở phần trước đó làm các đặc trưng cho hệ thống, do đó biểu thị cho độ lớn của khoảng tần số thứ *p* trong khung âm thanh tại thời điểm *t.*

Mục tiêu của RNN là chuyển đổi đầu vào thành chỗi ký tự cho nhãn tương ứng. Với mỗi khoảng thời gian *t,* RNN cho ra một xác suất dự đoán trên tập hợp các ký tự: với là một ký tự trong bảng chữ cái hoặc khoảng trắng, nháy đơn,…

Mô hình RNN mà nhóm dự kiến xây dựng bao gồm một số lớp đơn vị ẩn (hidden unit), một hoặc nhiều lớp phức hợp (convolution layer), tiếp đó là một hoặc nhiều lớp hồi quy (recurrent layer), cuối cùng là một hoặc nhiều lớp kết nối đầy đủ (connected layer).

Phần đầu tiên của mạng là một hoặc nhiều vòng chập theo thời gian của đầu vào. Đoạn âm thanh sau khi được tiền xử lí sẽ được đưa vào lớp này. Cửa sổ ngữ cảnh (Context window) là số lượng từ được dự đoán có thể xuất hiện trong phạm vi của từ đã cho (Hình 3.2). Đối với cửa sổ ngữ cảnh (context window) có kích thước *c*, nút kích hoạt thứ *i* ở bước thời gian *t* của lớp tích chập được đưa ra bởi công thức :

).

Trong đó: biểu thị yếu tố khôn ngoan giữa bộ lọc thứ *i* và của sổ ngữ cảnh của các lớp kích hoạt trước đó và hàm *f* là một hàm phi tuyến tính bậc nhất.

Hàm chỉnh lưu tuyến tính (ReLU) được rút gọn thành làm độ phi tuyến tính cho mô hình. Lấy mẫu phụ là một kỹ thuật đã được phát minh ra để giảm sự phụ thuộc vào định vị chính xác trong các danh sách các ánh xạ đặc trưng được tạo ra bởi các lớp phức hợp trong CNN. Mẫu phụ là một trong bộ nhỏ () được chọn bằng cách lấy ngẫu nhiên một cách đơn giản từ mẫu được sử dụng để bắt đầu lặp lại trên mẫu đó (. Nhóm sinh viên dự kiến lấy mẫu phụ bằng cách tách một số khung tính chập trong một số lớp, thường là lớp đầu tiên. Mục đích của việc này là rút ngắn số bước thời gian cho các lớp hồi quy ở trên.

Theo sau lớp tích tụ là một hoặc nhiều lớp hồi quy hai chiều. Ở mỗi lớp hồi quy hai chiều đều có tham số chuẩn hóa, trừ lớp đầu tiên. Đơn vị biểu diễn ẩn tại lớp được đặt là với quy ước biểu diễn cho đầu vào . Các lớp hồi quy kích hoạt tiến theo thời gian ( và lùi theo thời gian () được tính bằng công thức:

Hai tập hợp các nút kích hoạt được cộng lại để tạo thành các đầu ra cho lớp .

Hàm có thể rút gọn bằng phép hồi quy:

(1)

Trong đó

: trọng số tại lớp đơn vị ẩn thứ l tương ứng của đầu vào

: trọng số của ma trận hồi quy tại lớp ẩn thứ l

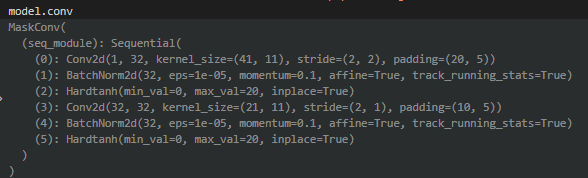
: độ sai lệch(bias) của lớp ẩn thứ l

Nhóm áp dụng một hoặc nhiều lớp kết nối đầy đủ tiếp theo sau các lớp hồi quy hai chiều, với

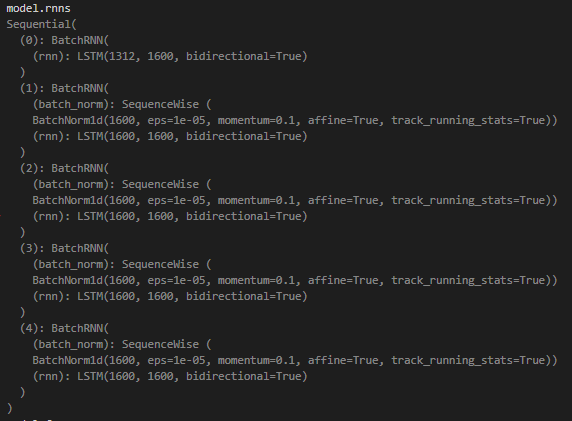
.

Đầu ra của lớp là một phép tính softmax phân bố xác suất trên các ký tự trong bộ kí tự của ngôn ngữ Tiếng Việt được cho bởi công thức:

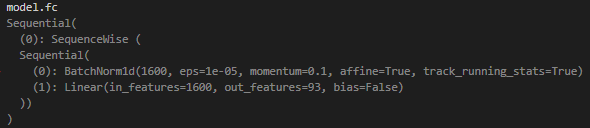
Minh họa các tham số của lớp kết nối đầy đủ, với mô hình RNN có 2 lớp tích tụ:



Tiếp theo lớp tích tụ là 5 lớp hồi quy hai chiều với kích thước mỗi lớp ẩn là 1600, tần số lấy mẫu (sample rate) của âm thanh đầu vào là 16000Hz, kích thước cửa sổ (windows size) là 0.02.



Cuối cùng là các lớp kết nối đầy đủ (fully connected layer) với kích thước mỗi lớp hồi quy hai chiều là 1600 và số lượng các kí tự của Tiếng Việt là 93:



Mô hình được huấn luyện bằng cách sử dụng hàm mất mát CTC (CTC loss function), với (x,y) là cặp giá trị đầu vào và đầu ra, là các tham số hiện tại của mạng.

Đạo hàm L theo các tham số của mạng được sử dụng đề cập nhật thông số của mạng thông qua sự lan truyền ngược (backpropagation) bằng thuật toán thời gian.

### Chuẩn hóa hàng loạt (Batch normalization)

Để mở rộng mô hình khi tăng thêm tập dữ liệu huấn luyện, nhóm sinh viên dự kiến tăng độ sâu của mạng bằng cách thêm nhiều lớp ẩn hơn thay vì làm cho các lớp lớn hơn. Chuẩn hóa hàng loạt là một kỹ thuật đào tạo các mạng nơ-ron theo chiều sâu, chuẩn hóa các đầu vào thành một lớp cho từng lô nhỏ. Mục đích chính của chuẩn hóa hàng loạt là giúp ổn định quá trình đào tạo và giảm số lượng epoch cần thiết để đào tạo mô hình.

Một phép biến đổi BatchNorm có công thức :

Trong đó:

* E : giá trị trung bình thực hiện trên một lô nhỏ.
* Var: phương sai thực nghiệm trên một lô nhỏ.
* : các tham số cho phép lớp chia tỉ lệ và dịch chuyển từng đơn vị ẩn như mong muốn, đây là các tham số có thể học được.
* : hằng số rất nhỏ dùng để ổn định số.

Trong các lớp tích chập, giá trị trung bình và phương sai được ước tính trên các đơn vị đầu ra tạm thơi cho một bộ lọc tích chập nhất định trên lô nhỏ.

Có hai phương pháp mở rộng BatchNorm cho RNN hai chiều:

Thêm một phép biến đổi BatchNorm ngay trước mọi trường hợp phi tuyến tính, phương trình g(.) (1) trở thành :

.

Trong phương pháp này, trung bình và phương sai được tích lũy qua một bước thời gian duy nhất của một lô nhỏ. Kỹ thuật này không dẫn để cải tiến trong việc tối ưu hóa.

Tính toán hồi quy theo công thức :

Qua mỗi đơn vị ẩn, trung bình và phương sai được tính toán trên tất cả các lô nhỏ theo chiều dài của chuỗi. Sử dụng kỹ thuật này, nhóm có thể đánh giá một lời nói duy nhất tại một thời điểm với kết quả tốt hơn so với đánh giá một loạt lớn đầu vào.

### SortaGrad

Khi huấn luyện các âm thanh mẫu có độ dài khác nhau đặt ra một số thách thức về thuật toán. Một giải pháp khả thi là cắt bớt sự lan truyền ngược theo thơi gian để tất cả mẫu có độ dài và trình tự giống nhau trong quá trình đào tạo. Tuy nhiên, điều này hạn chế khả năng học tập dài hạn cho mô hình.

Hàm chi phí CTC (CTC Cost Function) mà nhóm sẽ sử dụng ngầm định phụ thuộc vào độ dài câu nói.

(2)

Trong đó, Align (x, y) là tập hợp tất cả các căn chỉnh có thể có của các ký tự của phiên mã y với các âm thanh đầu vào x dưới toán tử CTC. Ví dụ: từ SUN có các Align S-UN, SU-N, -SUN,…

Trong phương trình (2), số hạng bên trong là tích theo các bước thời gian của dãy, số hạng này thu hẹp lại theo độ dài của dãy vì . là tích các nên < 1, do đó có thể nhỏ hơn 1, -log của tổng này là đầu ra của phương trình.

thúc đẩy chương trình đào tạo được gọi là SortaGrad, sử dụng độ dài của câu làm kinh nghiệm cho độ khó, vì các câu nói dài có chi phí cao hơn những câu nói ngắn.

### Biến đổi tần số (Freequency Convolution)

Tích chập tạm thời (Temporal convolution) thường được sử dụng trong nhận dạng giọng nói để mô hình hóa hiệu quả theo thời gian cho các phát âm có độ dài thay đổi. Nhiều mô hình giọng nói có lớp đầu tiên xử lý các khung đầu vào với một cửa sổ ngữ cảnh. Ngoài ra, lấy mẫu phụ là điều cần thiết để mạng nơ-ron hồi quy có thể kiếm soát tính toán với âm thanh có tốc độ mẫu cao (hign sample-rate).

Các biến đổi trong miền tần số và thời gian khi áp dụng cho các đặc trưng đầu vào trước bất kỳ quá trình xử lý nào khác có thể cải thiện hiệu suất ASR. Sự biến đổi tần số cố gắng mô hình hóa phương sai do dự biến đổi của loa ngắn hơn so với các mạng kết nối đầy đủ lớn. Vì thứ tự âm phổ của các đặc trưng bị loại bỏ bởi các lớp kết nối đầy đủ và lớp hồi quy, nên biến đổi tần số hoạt động tốt hơn trên các lớp đầu tiên của mạng.

### Bước sóng (Striding)

Trong các lớp tích chập, áp dụng bước sóng dài hơn và bối cảnh (context) rộng hơn để tăng tốc quá trình đào tạo vì chỉ cần ít bước thời gian hơn để mô hình hóa một đầu vào. Việc lấy âm thanh mẫu đầu vào thông qua FFT và bước sóng chập làm giảm số lượng bước thời gian và tính toán cần thiết trong các lớp, nhưng làm giảm hiệu suất.

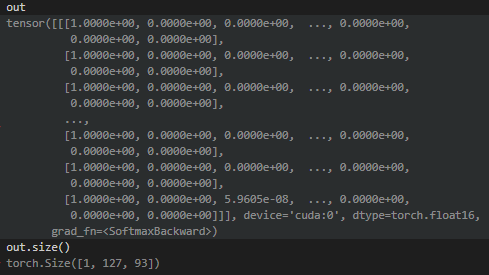
## Giải pháp xây dựng bộ giải mã

Để đưa các từ ra khỏi một mô hình được đào tạo, người ta cần sử dụng bộ giải mã. Bộ giải mã chuyển đổi phân phối xác suất của ma trận kết quả trên các ký tự thành văn bản Tiếng Việt cuối cùng. Có hai loại bộ giải mã thường được sử dụng với các mô hình dựa trên CTC: bộ giải mã tham lam và bộ giải mã tìm kiếm chùm với tính điểm lại mô hình ngôn ngữ.

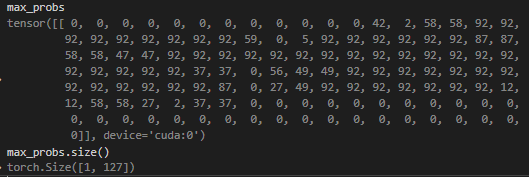
### Bộ giải mã tham lam

Một bộ giải mã tham lam xuất ra ký tự có thể xảy ra nhất ở mỗi bước thời gian. Nó rất nhanh và nó có thể tạo ra các bản ghi rất gần với cách phát âm gốc. Nhưng nó có thể mắc nhiều lỗi chính tả nhỏ.

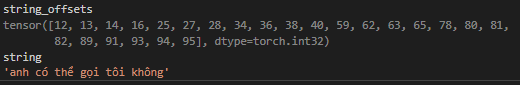
Nhóm sinh viên sử dụng câu nói: “anh có thể gọi tôi không” để dự đoán văn bản đầu ra của mô hình với bộ giải mã tham lam. Mô hình được huấn luyện có 5 lớp hồi quy hai chiều, mỗi lớp có 1600 lớp đơn vị ẩn và được huấn luyện với LSTM. Một output của mô hình sau khi qua mạng nơ – ron được một ma trận. Một ma trận 3 chiều thể hiện phân phối xác suất của từng kí tự trên mỗi bước thời gian.



Bộ giải mã tham sẽ chọn những kí tự có xác suất cao nhất trên mỗi bước thời gian và trả về ma trận là thứ tự các kí tự dự đoán trong bảng chữ cái Tiếng Việt của văn bản đầu ra:



Sau khi có được ma trận thứ tự các kí tự, nhóm dựa vào bảng chữ cái Tiếng Việt để ánh xạ thành văn bản hoàn chỉnh.



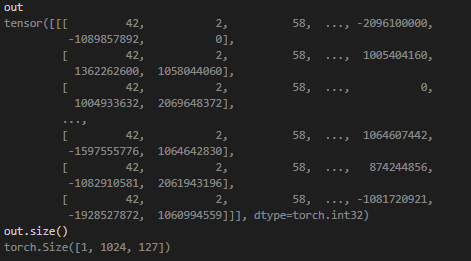
### Bộ giải mã tìm kiếm chùm với mô hình ngôn ngữ

Bộ giải mã tham lam đem lại tốc độ nhanh cho quá trình đào tạo nhưng nó có thể mắc nhiều lỗi chính tả nhỏ. Hơn thế nữa, do bản chất của chỉ số WER, ngay cả một lỗi ký tự cũng làm cho cả một từ không chính xác. Một bộ giải mã tìm kiếm chùm với tính điểm lại mô hình ngôn ngữ cho phép kiểm tra nhiều giải mã có thể cùng một lúc với việc ấn định điểm cao hơn cho nhiều N-gram có thể xảy ra hơn theo một mô hình ngôn ngữ nhất định. Việc tích hợp chung với mô hình ngôn ngữ giúp sửa lỗi chính tả. Tuy nhiên, nhược điểm là nó chậm hơn đáng kể so với một bộ giải mã tham lam.

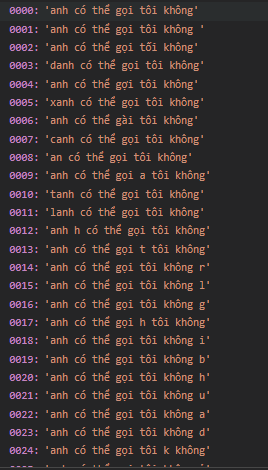
Nhóm dự kiến sử dụng thuật toán tìm kiếm chùm (BeamSearch) để tìm phiên âm tối ưu. Ý tưởng của BeamSearch là tìm kiếm từ có xác suất xuất hiện cao nhất, và từ này sẽ là vector đầu vào để dự đoán từ tiếp theo.

Thuật toán BeamSearch có tham số gọi là beam width, tại mỗi bước dự đoán, thay vì chọn từ xác suất lớn nhất, thuật toán chọn beam witdh kết quả có xác suất cao nhất. Tiếp tục tính toán các bộ xác suất cho đến khi kết thúc, ta sẽ thu được một cây xác xuất. Nhân các xác suất của cả chuỗi có điều kiện, câu kết quả là câu có xác xuất lớn nhất.

Sử dụng ma trận kết quả của ví dụ ở bộ giải mã tham lam, ma trận phân phối xác suất này sẽ được áp dụng thuật toán tìm kiếm chùm với beam width là 1024 và cho ra kết quả:



Sau khi có được ma trận thứ tự các kí tự trên từng bước thời gian, thuật toán tìm kiếm chùm dựa vào bảng chữ cái Tiếng Việt để ánh xạ thành văn bản hoàn chỉnh cho từng kết quả trong beam width.



Cuối cùng nhóm chọn kết quả trả về đầu tiên trong tất cả kết quả làm kết quả cuối cùng của mô hình khi sử dụng thuật toán tìm kiếm chùm.

### Mô hình ngôn ngữ (Language model)

Mô hình RNN được đào tạo qua bộ dữ liệu lớn. Để mạng có thể thành thạo chính ram không có ràng buộc về ngôn ngữ bên ngoài, cải thiện WER, nhóm dự kiến sử dụng mô hình ngôn ngữ n-gram (cụ thể là 5-gram) vì chúng mở rộng quy mô phù hợp với lượng lớn văn bản không được đánh nhãn.

Mô hình 5-grams của câu nói “bạn cho tôi mượn được không”

\1-grams:

-1.20412    <unk>   0

0   <s> -0.30103

-0.87312675 </s>    0

-0.87312675 bạn -0.30103

-0.87312675 cho -0.30103

-0.87312675 tôi -0.30103

-0.87312675 mượn    -0.30103

-0.87312675 được    -0.30103

-0.87312675 không   -0.30103

\2-grams:

-0.24644431 không </s>  0

-0.24644431 <s> bạn -0.30103

-0.24644431 bạn cho -0.30103

-0.24644431 cho tôi -0.30103

-0.24644431 tôi mượn    -0.30103

-0.24644431 mượn được   -0.30103

-0.24644431 được không  -0.30103

\3-grams:

-0.105970904    được không </s> 0

-0.105970904    <s> bạn cho -0.30103

-0.105970904    bạn cho tôi -0.30103

-0.105970904    cho tôi mượn    -0.30103

-0.105970904    tôi mượn được   -0.30103

-0.105970904    mượn được không -0.30103

\4-grams:

-0.049761247    mượn được không </s>    0

-0.049761247    <s> bạn cho tôi -0.30103

-0.049761247    bạn cho tôi mượn    -0.30103

-0.049761247    cho tôi mượn được   -0.30103

-0.049761247    tôi mượn được không -0.30103

\5-grams:

-0.024168313    tôi mượn được không </s>

-0.024168313    <s> bạn cho tôi mượn

-0.024168313    bạn cho tôi mượn được

-0.024168313    cho tôi mượn được không

\end\

Trong quá trình tìm hiểu, phiên âm y tối đa hóa Q(y):

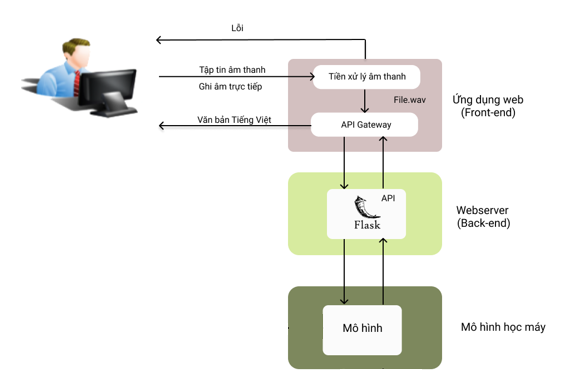
(3)

Trọng số kiểm soát các đóng góp tương đối của mô hình ngôn ngữ và mạng CTC.

Trọng số khuyến khích nhiều từ hơn trong phiên âm. Các tham số này được điều chỉnh nhiều hơn trong một tập phát triển.

## Giải pháp xây dựng máy chủ

Nhóm sinh viên dự kiến xây dựng máy chủ với mục đích tạo ra giao diện lập trình ứng dụng (API), làm trung gian giữa ứng dụng mẫu và mô hình âm thanh đã được huấn luyện.



Máy chủ cho phép ứng dụng mẫu gửi đến một đoạn âm thanh dạng wav của tiếng Việt, sử dụng mô hình học máy xử lý đoạn âm thanh và trả về đoạn văn bản tiếng Việt cho ứng dụng mẫu.

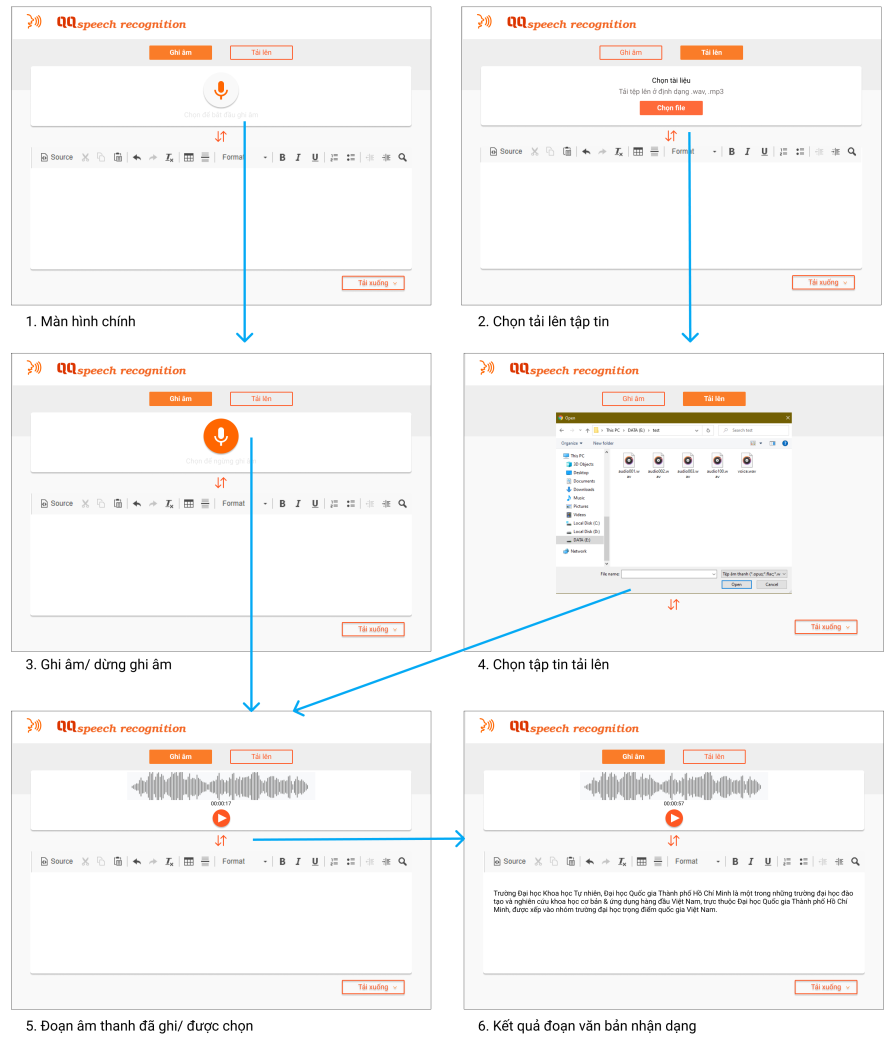
Nhóm lựa chọn sử dụng Framework Flask của python để xây dựng máy chủ. Vì Flask dễ cài đặt và triển khai, phù hợp với ứng dụng web quy mô vừa và nhỏ, giúp tập trung phát triển mục tiêu của nhóm là xây dựng giao diện lập trình ứng dụng, dễ dàng triển khai trên các dịch vụ máy chủ đám mây.

Sau khi xây dựng máy chủ, nhóm dự kiến triển khai máy chủ lên nền tảng đám mây Heroku.

## Giải pháp xây dựng ứng dụng mẫu

Theo giải pháp về hệ thống, nhóm sinh viên xây dựng một ứng dụng mẫu với nhiệm vụ cho phép người dùng sử dụng dịch vụ chuyển đổi âm thanh tiếng Việt thành văn bản tiếng Việt. Ứng dụng mẫu sử dụng giao diện lập trình ứng dụng (API) của máy chủ vừa được tạo ở trên.

Bản mẫu giao diện ứng dụng:



# QUÁ TRÌNH HUẤN LUYỆN MÔ HÌNH

## Mô hình huấn luyện được sử dụng

* Mô hình dự kiến huấn luyện chính: DeepSpeech 2 dựa theo bài báo *Deep Speech 2: End-to-End Speech Recognition in English and Mandarin* (<https://arxiv.org/pdf/1512.02595.pdf>), dựa trên 2 nguồn hướng dẫn cài đặt
  + <https://github.com/SeanNaren/deepspeech.pytorch>
  + <https://github.com/tensorflow/models/tree/master/research/deep_speech>
* Mô hình dự kiến chọn để so sánh: DeepSpeech 0.8.1 dựa theo bài báo *Deep Speech: Scaling up end-to-end speech recognition* (<https://arxiv.org/abs/1412.5567>) , dựa trên nguồn hướng dẫn:
  + <https://deepspeech.readthedocs.io/en/v0.8.2/>

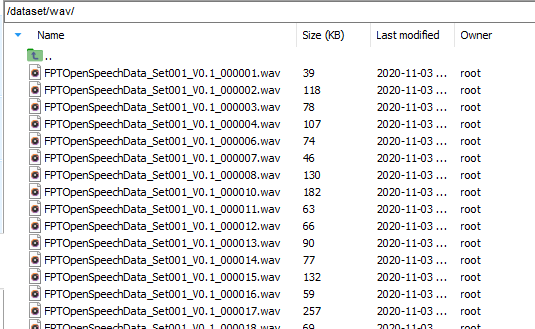
## dữ liệu huấn luyện mô hình

Để huấn luyện một mô hình nhận dạng âm thanh tiếng Việt dựa trên DeepSpeech2 thì bộ dữ liệu phải có ba thành phần chính bao gồm:

* Âm thanh

Để huấn luyện một hệ thống nhận dạng âm thanh tiếng Việt đủ tốt thì lượng dữ liệu âm thanh dùng để huấn luyện cũng phải đủ nhiều và đủ tốt. Ngoài ra dữ liệu âm thanh dùng để huấn luyện cũng cần thoả mãn các điều kiện:

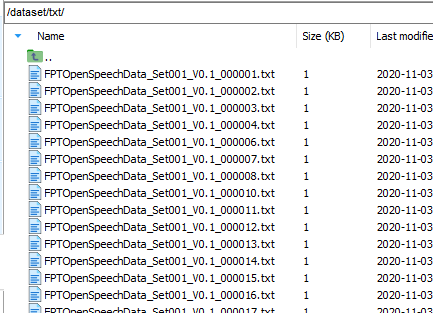
* Độ dài tối đa của mỗi tập tin âm thanh là 10 giây.
* Tập tin âm thanh cần chuyển về định dạng \*.wav nhằm đồng nhất định dạng đầu vào và tăng chất lượng âm thanh.
* Giá trị số lần lấy mẫu trên 1 giây (sample rate) là 16000 Hz.



* Văn bản

Là bản dịch (transcript) tương ứng với nội dung lời nói trong các tập tin âm thanh.

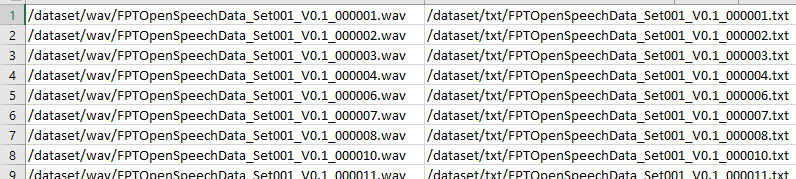
Dữ liệu âm thanh và văn bản sau khi đã được kiểm chứng và sàn lọc sẽ được tổng hợp trong một thư mục txt, thư mục này chứa các văn bản Tiếng Việt của từng âm thanh mẫu.



* Tập tin có định dạng \*.csv

Tập tin này tổng hợp đường dẫn tập tin âm thanh mẫu và tập tin văn bản với nội dung được chia thành 3 cột theo thứ tự sau:

* Cột thứ nhất: Nội dung cột này chứa đường dẫn trực tiếp đến một tập tin âm thanh.
* Cột thứ hai: Nội dung cột này chứa đường dẫn trực tiếp đến một tập tin văn bản tương ứng của tập tin âm thanh.



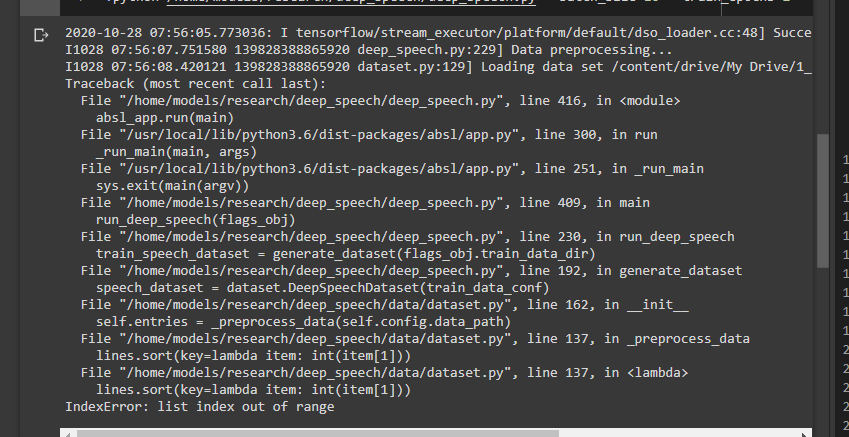
## Xây dựng language model

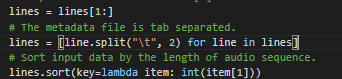
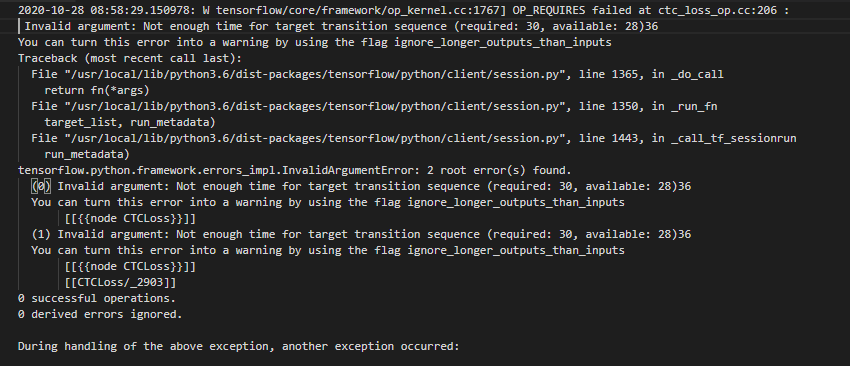
Nhằm tăng độ chính xác cho mô hình được sử dụng, nhóm sinh viên tự xây dựng mô hình ngôn ngữ phục vụ riêng cho mô hình nhận dạng. Dữ liệu cho mô hình ngôn ngữ được nhóm sinh viên chọn lọc và lấy từ các trang báo điện tử, các trang truyện ngắn và chọn lọc lại các từ thích hợp trong từ điển do nhóm tự xây dựng trước đó.

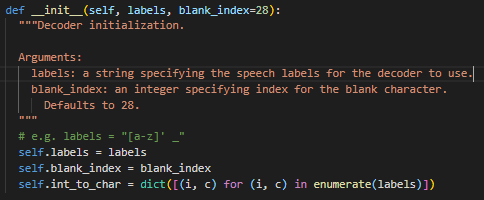
## Các lỗi huấn luyện

### Mô hình DeepSpeech 2

* Huấn luyện mô hình Deepspeech2 với Tiếng Việt theo hướng dẫn: <https://github.com/tensorflow/models/tree/master/research/deep_speech>
* List index out of range



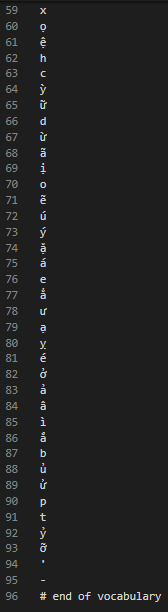
* Nguyên nhân : định dạng nội dung trong file .csv không phù hợp 
* Giải pháp : sửa ‘/t’ thành ‘,’
* Lỗi Invalid argument: Not enough time for target transition sequence  
  
* Nguyên nhân:
* Giải pháp: xóa các file lỗi
* Lỗi vocabulary của Tiếng Việt



Nguyên nhân: Trong file vocabulary của Tiếng Việt, Tiếng Anh vị trí kí tự ‘-‘ khác nhau

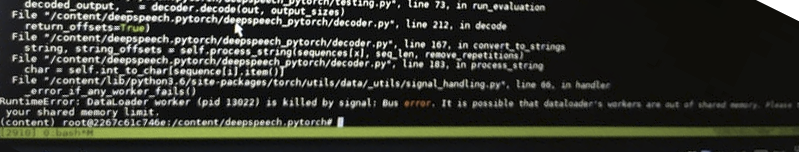
Trong file vocabulary của Tiêng Anh blank\_index =28 (kí tự ‘-‘ nằm ở dòng 29)

Trong file vocabulary của Tiêng Việt blank\_index =91 (kí tự ‘-‘ nằm ở dòng 92)

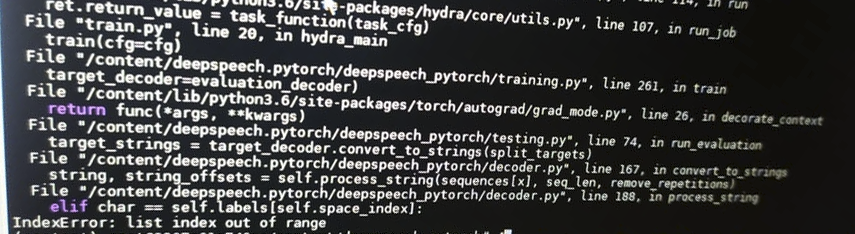


Hướng dẫn sử dụng của mô hình không chỉ rõ cách để test 1 file .wav sau khi train thành công mô hình.

* Huấn luyện mô hình Deepspeech2 với Tiếng Việt theo hướng dẫn: <https://github.com/SeanNaren/deepspeech.pytorch>
* ERROR: Unexpected bus error encountered in worker. This might be caused by insufficient shared memory (shm)



* Nguyên nhân : không đủ bộ nhớ chia sẻ trên RAM
* Cách khắc phục : Giảm batch\_size xuống 32, giảm data.num\_workder = 0 hoặc 1
* List index out of range



* Nguyên nhân : vượt quá phạm vi của mảng trong code ( labels[n] với n là độ dài mảng)
* Cách khắc phục : sửa thành label[n-1]
* Lỗi Unicode

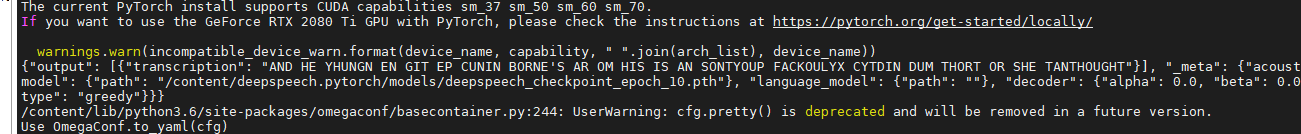


* Nguyên nhân : hệ điều hành không nhận diện được Unicode
* Cách khắc phục :   
  export LC\_ALL="en\_US.UTF-8"

export LC\_CTYPE="en\_US.UTF-8"

sudo dpkg-reconfigure locales

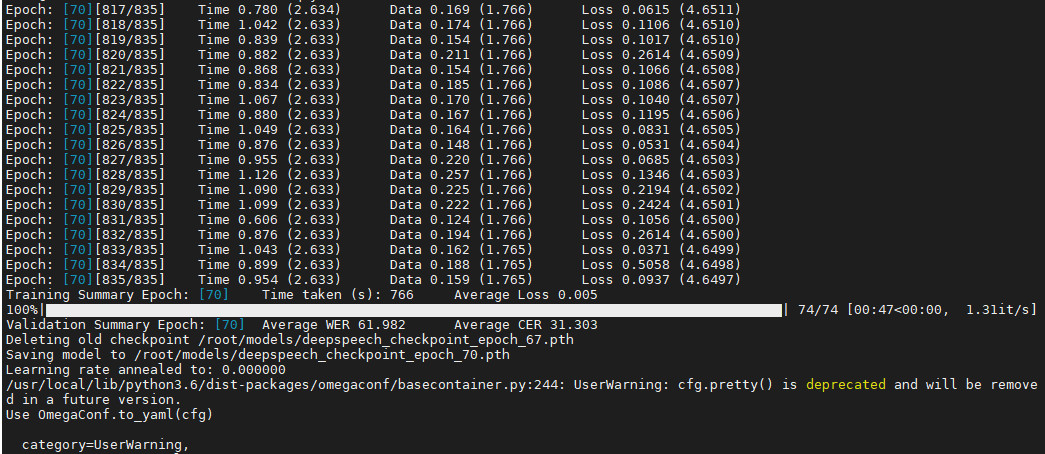
* Warning : không tương thích giữa Geforce RTX và Pytorch



* Nguyên nhân : không tương thích giữa phiên bản Cuda và phiên bản Pytorch
* Cách khắc phục: cài đặt phiên bản tương thích   
  pip install torch==1.7.0+cu101 torchvision==0.8.1+cu101 torchaudio===0.7.0 -f <https://download.pytorch.org/whl/torch_stable.html>

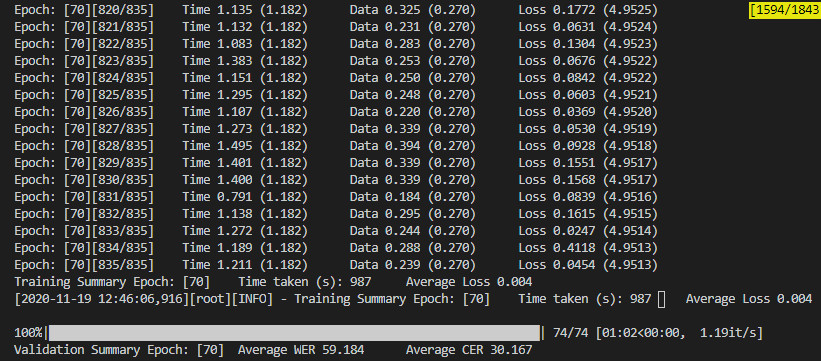
## Kết quả huấn luyện

* Lần chạy huấn luyện đầu tiên với batch size 70, bộ dataset FPT, hidden size là 1024



Tỉ lệ lỗi từ 61.982

* Lần chạy huấn luyện đầu tiên với batch size 70, bộ dataset FPT, hidden size là 512



Tỉ lệ lỗi từ 59.184

* Bảng tổng hợp một số thay đổi tham số của quá trình huấn luyện mà nhóm đã thực hiện:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ngày | Train manifest | Val manifest | rnn\_type | epochs | Batch size | Hidden size | Hidden layers | WER | Training Time |
| 6/11/2020 | vi\_train | vi\_test | lstm | 75 | 32 | 1024 | 7 | 61.982 |  |
| 19/11/2020 | vi\_train | vi\_test | lstm | 75 | 32 | 512 | 7 | 59.184 |  |
| 9/12/2020 | vi\_train | vi\_test | gru | 50 | 16 | 1600 | 5 | 53.550 | 3 ngày |
| 14/12/2020 | vi\_train | vi\_test | gru | 70 | 16 | 1600 | 5 | 58.060 | 3 ngày |
| 17/12/2020 | vi\_train | vi\_test | lstm | 50 | 16 | 1600 | 5 | 55.729 | 3 ngày |