ĐẠI HỌC QUỐC GIA TP HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA

BÁO CÁO ĐỒ ÁN TỔNG HỢP

Tên đề tài

ASR DATA ARGUMENTATION

GVHD: Võ Thanh Hùng
Sinh viên: Phạm Anh Dũng - 2110101
Lớp: L06

TP.Hồ Chí Minh, Tháng 11 năm 2023

Họ và tên: Phạm Anh Dũng

MSSV: 2110101

Kế hoạch thực hiện

Tuần	Nội dung	Ghi chú thêm
Tuần 1	Literature review	
Tuần 2	Literature-review	
Tuần 3	Viết Project Summary	
Tuần 4	Tìm hiểu về các mô hình ASR và các phương pháp augment data	
Tuần 5	Chạy được clone code ASR	
Tuần 6	Chạy được clone code augment data	
Tuần 7	Tìm kiếm data thích hợp	
Tuần 8	Code thực thi ASR	
Tuần 9	Code thực thi augment data	
Tuần 10	Tối ưu và sửa lỗi code	
Tuần 11	Thực hiện với các data khác nhau để so sánh	
Tuần 12	Thực hiện demo đề tài	
Tuần 13	Thực hiện viết báo cáo cuối kì	
Tuần 14	Chỉnh sửa, hoàn thiện báo cáo và demo	

Weekly report from 1/10 - 7/10:

Những nội dung đã được thực hiện trong tuần

- Tìm hiểu Mạng nơ-ron nhân tạo (ANN), mạng nơ-ron hồi quy (RNN), LSTM
- Tìm hiểu Word Embedding (CBOW model, Skip-gram model), Language Model (N-gram, Structured)

Những vấn đề khó khăn đang gặp phải

- Còn nhiều thuật ngữ chuyên ngành cần tìm hiểu thêm (softmax, Convolution, kernel, cross-entropy, gradient descent ...)
- Các công thức đại số, toán học còn khó hiểu

Kế hoạch của tuần tiếp theo

Tìm kiếm thêm tài liệu, tìm hiểu thêm về các từ khoá chưa hiểu.

Weekly report from 8/10 - 14/10:

Những nội dung đã được thực hiện trong tuần

- Tìm hiểu về mô hình ASR Deep Speech 2
- Tìm hiểu phương pháp xử lý dữ liệu âm thanh đầu vào (Fast Fourier Transform, Spectogram) và dữ liệu văn bản cho mô hình ngôn ngữ
- Tìm hiểu phương pháp xây dựng Language Model, tham khảo từ bài luận văn "Kết hợp học sâu và mô hình ngôn ngữ để nhận dạng giọng nói tiếng Việt" của Nguyễn Đức Huy, Đại học Bách Khoa TP.HCM. Link tài liệu: https://opac.lib.hcmut.edu.vn/search/detail.asp?aID=1&ID=31290&fbclid=IwAR3oB3e8B2S4EUPoCGGrgkbt8NFzdZyCpg2xaVZRoRymEmfAbdsWUasp3rg

Những vấn đề khó khăn đang gặp phải

- Chưa gặp khó khăn gì đáng kể, các tài liệu mô tả khá trực quan dễ hiểu

Kế hoạch của tuần tiếp theo

- Đọc tài liệu hướng dẫn sử dụng phương Wav2Vec2 trên tập dữ liệu MInDS-14.

Weekly report from 15/10 - 21/10:

Những nội dung đã được thực hiện trong tuần

- Làm quen với cách xử lí các input dữ liệu âm thanh và train một mô hình ASR đơn giản sử dụng thư viện tensorflow theo bài hướng dẫn của website Tensorflow. Link bài viết : https://www.tensorflow.org/tutorials/audio/simple_audio
- Finetune Wav2Vec2 trên tập dữ liệu MInDS-14 theo hưỡng dẫn của bài viết trên Hugging Face của tác giả Wauplin. Link bài hướng dẫn : https://github.com/huggingface/transformers

Những vấn đề khó khăn đang gặp phải

- Hình dung được các thao tác cần làm cho việc huấn luyện dữ liệu âm thanh qua việc thực hành train mô hình ASR đơn giản, tuy nhiên mô hình được train còn thiếu chính xác.
- Chưa thể thực hiện được finetune mô hình Wav2Vec2, không train được mô hình do trục trặc trong xử lý lỗi code. Còn nhiều thứ cần làm quen khi xử lý mô hình AI bằng Python

Kế hoạch của tuần tiếp theo

- Tìm hiểu thêm về xử lý mô hình AI bằng Python, đọc thêm các tài liệu tham khảo

Weekly report from 22/10 - 28/10:

Những nội dung đã được thực hiện trong tuần

- Tìm hiểu về Anaconda, thực hiện train model trên môi trường ảo.
- Train model trên local với dataset loading từ hugging face, link bài hướng dẫn https://github.com/dreji18/Fine-tune-Speech-Recognition/blob/main/data%20preparation.py

Những vấn đề khó khăn đang gặp phải

- Vì dataset lớn và các thư viện phải sử dụng nhiều thư viện nên việc train mô hình trên local không khả thi, ngoài ra gặp nhiều trục trặc lỗi về xung đột phiên bản các thư viện, các lỗi import khi thiết lập môi trường ảo bằng anaconda.
- Chưa phân bổ thời gian hợp lí với các bài tập lớn các môn trên trường dẫn đến chưa thực hiện đầy đủ nội dung tuần hiện tại

Kế hoạch của tuần tiếp theo

- Quay trờ lại thực hiện việc train model trên cloud sử dụng google colab, đọc thêm các tài liệu tham khảo.

Weekly report from 29/10 - 4/11:

Những nội dung đã thực hiện được trong tuần

- Tìm hiểu và sử dụng Neural Modules (NeMo) toolkit, thực hiện train mô hình trên google colab, link bài hướng dẫn https://colab.research.google.com/github/NVIDIA/NeMo/blob/stable/tutorials/asr/ASR_with_NeMo.ipynb#scrollTo=inRJsnrz1psq

Những vấn đề khó khăn đang gặp phải

- Không gặp khó khăn gì đáng kể khi làm theo hướng dẫn

Kế hoạch của tuần tiếp theo

- Tìm hiểu tiếp về phương pháp sử dụng toolkit NeMo ở bài hướng dẫn trên, thử với các datasets khác và tìm hiểu áp dụng data argument trên mô hình này.

Weekly report from 5/11 - 11/11:

Những nội dung đã thực hiện được trong tuần

- Tìm kiếm các data phù hợp để sử dụng train cho mô hình kết hợp với NeMo toolkit
- các dataset phù hợp:
  https://huggingface.co/datasets/speech_commands
  https://huggingface.co/datasets/mozilla-foundation/common_voice_11_0/tree/main
  https://github.com/Jakobovski/free-spoken-digit-dataset

Những vấn đề khó khăn đang gặp phải

- Chỉ đang tiếp cận với các data ngôn ngữ Anh chưa xử lí các ngôn ngữ khác
- Đa số các dataset trên có dung lượng lớn, train mất nhiều thời gian

Kế hoạch của tuần tiếp theo

- Thực hiện train trên dataset Speech Command và so sánh khi thực hiện Spec augmentation.

Weekly report from 12/11 - 18/11:

Những nội dung đã thực hiện được trong tuần

- Train thành công mô hình trên dataset Speech Command. Thực hiện so sánh khi có và không có Data Augmentation

Những vấn đề khó khăn đang gặp phải

- Gặp vài khó hiểu trong việc thực hiện config cho mô hình (chọn số epoch, batch size, ...)

Kế hoạch của tuần tiếp theo

- Thực hiện train mô hình trên bộ dữ liệu LibriSpeech.

Weekly report from 19/11 - 25/11:

Những nội dung đã thực hiện được trong tuần

- Train mô hình trên tập dữ liệu LibriSpeech dev-set và kiểm tra trên tập test-set.

Những vấn đề khó khăn đang gặp phải

- Kết quả val_wer còn kém hiệu quả (0.9). Đang tìm cách khắc phục

Kế hoạch của tuần tiếp theo

- Thực hiện viết báo cáo cuối kì, chỉnh sửa bổ sung mô hình.

Weekly report from 26/11 - 2/12:

Những nội dung đã thực hiện được trong tuần

- Thực hiện viết báo cáo cuối kì, điều chỉnh lại mô hình (thời lượng, số câu thoại, thông số trong config, ...)

Những vấn đề khó khăn đang gặp phải

- Huấn luyện mô hình trên google Colab bị giới hạn về số epoch do đạt giới hạn dung lượng GPU. Cân nhắc đổi sang các nền tảng khác (Kaggle) hoặc điều chỉnh lại các thông số huấn luyện.

Kế hoạch của tuần tiếp theo

- Tiếp tục chỉnh sửa, bổ sung hoàn thành báo cáo. Thực hiện so sánh giữa các điều chỉnh với mô hình và trên các tập dữ liệu khác nhau.

Weekly report from 2/12 - 8/12

Những nội dung đã thực hiện được trong tuần

- Tiếp tục hoàn thiện báo cáo cuối kì

Những vấn đề khó khăn đang gặp phải

- Không có khó khăn nào đáng kể

Kế hoạch của tuần tiếp theo

- Tiếp tục chỉnh sửa, bổ sung hoàn thành báo cáo, huấn luyện mô hình.

Weekly report from 9/12 - 15/12

Những nội dung đã thực hiện được trong tuần

- Tiếp tục hoàn thiện báo cáo cuối kì

Những vấn đề khó khăn đang gặp phải

- Không có khó khăn nào đáng kể

Kế hoạch của tuần tiếp theo

- Tiếp tục chỉnh sửa, bổ sung hoàn thành báo cáo, huấn luyện mô hình.

Weekly report from 16/12 - 22/12

Những nội dung đã thực hiện được trong tuần

- Hoàn thiện báo cáo

Những vấn đề khó khăn đang gặp phải

- Không có khó khăn nào đáng kể

Kế hoạch của tuần tiếp theo

MỤC LỤC

Danh sách hình vẽ

1 Cơ sở lý thuyết

1.1 Tổng quan về ASR

Trong cuộc sống có nhiều trường hợp ta cần chuyển giọng nói con người thành văn bản để có thể xử lý tốt hơn, ví dụ như ghi chép nội dung trong cuộc họp, tạo bản ghi nhớ, tạo phụ đề cho video,... bài toán trên được gọi là bài toán Speech to text. Từ đó chúng ta có hệ thống ASR, một hệ thống ASR (Automatic Speech Recognition) là hệ thống trả về các câu văn bản có xác suất khớp nhất với tín hiệu âm thanh giọng nói đầu vào. Việc sử dụng phương pháp tiếp cận thống kê (Statistical Approach) trong lĩnh vực nghiên cứu ASR đã đem lại nhiều kết quả đáng ghi nhận.
Như đã đề cập ở trên, ASR trả về câu văn bản có xác suất cao nhất $\widehat{H}$, từ các đầu vào âm thanh là chuỗi các giá trị quan sát O cho trước:

$$\widehat{H} = \arg\max_{H} P\left(H|O\right)$$

Sử dụng công thức *Bayes*, ta có thể biến đổi lại phương trình trên như sau:

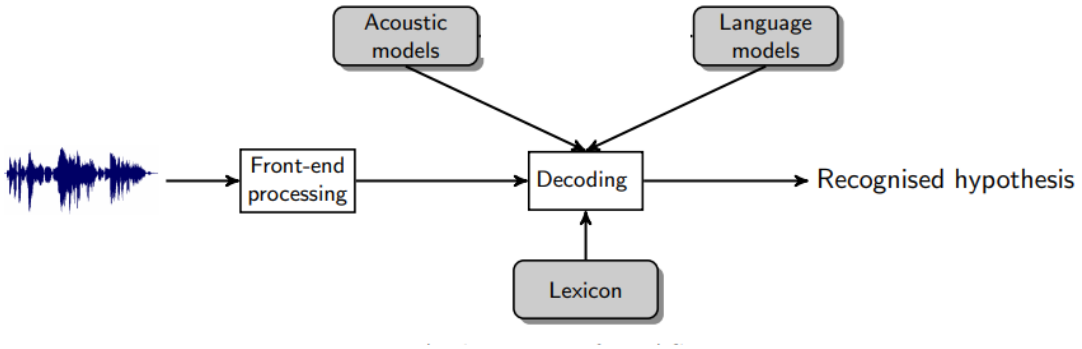
$$\widehat{H} = \arg\max_{H} P\left(H|O\right) = \arg\max_{H} \frac{P\left(O|H\right)P\left(H\right)}{P\left(O\right)} = \arg\max_{H} P\left(O|H\right)P\left(H\right)$$

Xác suất $P\left(O\right)$ không thay đổi với mọi H nên ta có thể bỏ đi khỏi phương trình và không ảnh hưởng đến việc xác định $\widehat{H}$. $P\left(O|H\right)$ được tính bởi mô hình âm thanh (Acoustic model) và $P\left(H\right)$ được tính bởi mô hình ngôn ngữ (Language model).

1.2 Kiến trúc hệ thống ASR

Hệ thống ASR có cấu tạo gồm các thành phần chính sau:

- **Feature Extraction**: Chuyển tín hiệu âm thanh thành một chuỗi các vector âm thanh đặc trưng. Các giá trị quan sát này được "*nén*" lại và mang đủ thông tin cần thiết để có thể nhận dạng được tại giai đoạn sau.
- **Acoustic Model**: Chứa các dữ liệu thống kê biểu diễn riêng biệt cho âm thanh tạo nên các từ trong Language Model.
- **Language Model**: Chứa một danh sách lớn các từ và xác suất xuất hiện của chúng trong chuỗi được cho.
- **Decoder**: Là một chương trình lấy các âm thanh từ giọng nói và tìm kiếm trong Acoustic Model các âm thanh tương đương. Khi tìm thấy một kết quả phù hợp, bộ giải mã xác định âm vị tương ứng với âm thanh đó. Nó tiếp tục lưu lại các âm vị phù hợp đã tìm được cho đến khi gặp một điểm dừng trong lời nói của người nói, từ đó tổng hợp thành một chuỗi âm vị. Sau đó tìm kiếm trong Language Model chuỗi âm vị tương đương. Nếu tìm thấy một kết quả phù hợp, nó sẽ trả về văn bản của từ hoặc cụm từ tương ứng.



Hình 1: Kiến trúc hệ thống ASR

1.3 Models

Có hai model chính trong ASR là: HMM-based model và End-to-End model

1.3.1 HMM-Based Model

HMM-Based Model là một mô hình phổ biến được sử dụng trong khoảng thời gian dài với độ chính xác cao. Trong HMM-Based Model, các mô hình độc lập và có các vai trò khác nhau. Acoustic model mô hình hoá các ánh xạ đầu vào lời nói với các chuỗi đặc trưng, Pronunciation model thì ánh xạ các âm vị với các grapheme (tự vị - đơn vị nhỏ nhất trong ngôn ngữ viết), Language model ánh xạ các chuỗi kí tự thành một câu hoàn chỉnh.

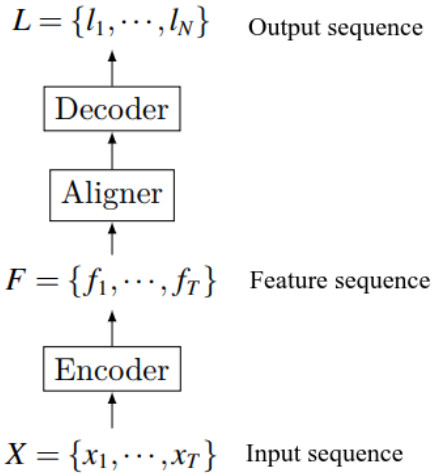
Acoustic Model : Trong acoustic model, xác suất quan sát $P(O|H)$ được tính bởi phương pháp GMM. Mặt khác, với phương pháp DNN (Deep Neurol Network) ta trực tiếp tính xác suất hậu nghiệm $P(H|O)$. Từ đó hình thành tên gọi của hai mô hình là HMM-GMM và HMM-DNN. Với sự phát triển mạnh mẽ của Deep Learning, DNN vượt trội hơn GMM và dần thay thế GMM trở thành phương pháp phổ biến trong acoustic model. Trong HMM-based model các module sử dụng kĩ thuật khác nhau và vài trò khác nhau, HMM chủ yếu được dùng để tính DTW (dynamic time warping) ở mức frame trong khi DNN và GMM được dùng để tính xác suất thực nghiệm (emission probability).

Hạn chế :

- Vì mỗi module trong HMM-based model sử dụng kĩ thuật khác nhau và vai trò khác nhau, gây ra khó khăn, phức tạp trong việc tối ưu toàn bộ mô hình.
- Để đơn giản hoá trong việc xây dựng và huấn luyện mô hình, HMM-based model giả định tính độc lập của các xác suất có điều kiện. Điều này không đúng trong thực tế, làm giảm tính chính xác của mô hình.

1.3.2 End-to-End Model

Với sự phát triển của Deep Learning, ngày càng nhiều các nghiên cứu trong việc ứng dụng end-to-end vào ASR.



Hình 2: Cấu trúc mô hình end-to-end

End-to-End model là mô hình ánh xạ trực tiếp các chuỗi âm thanh đầu vào thành các chuỗi từ đầu ra. Cấu tạo của hầu hết các End-to-End model bao gồm: Encoder mã hoá chuỗi đầu vào thành chuỗi đặc trưng, Aligner thực hiện align giữa chuỗi đặc trưng và ngôn ngữ, Decoder giải mã và cho ra kết quả cuối cùng. Việc phân chia này không phải lúc nào cũng rõ ràng vì end-to-end là một cấu trúc hoàn chỉnh. Ngược lại với mô hình dựa trên HMM, bao gồm nhiều module, mô hình end-to-end thay thế các module bằng một mạng nơ-ron, thực hiện ánh xạ trực tiếp tín hiệu âm thanh thành các chuỗi label mà không cần các trạng thái trung gian được thiết kế chi tiết. Ngoài ra không cần thực hiện xử lý hậu kỳ trên kết quả đầu ra.

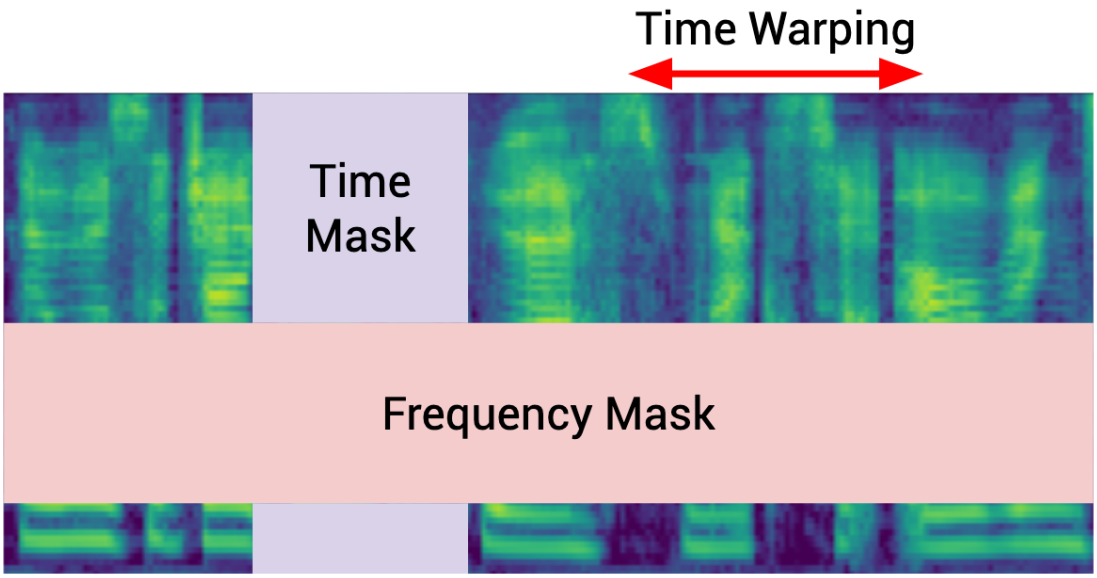
Các mô hình end-to-end chủ yếu được chia thành ba loại khác nhau dựa trên cách triển khai:

- **CTC-based**: Ban đầu mô hình sẽ liệt kê tất cả các hard alignment có thể. Sau đó sẽ đạt được soft alignment bằng cách tổng hợp các hard alignment. Mô hình giả định rằng các output label là độc lập với nhau khi liệt kê hard alignment.
- **RNN-transducer**: Mô hình cũng liệt kê tất cả các hard alignment có thể xảy ra và sau đó tổng hợp chúng để có soft alignment. Nhưng không giống như CTC, RNN-transducer không giả định các output label độc lập khi liệt kê các hard alignment.
- **Attention-based**: Phương pháp này không liệt kê tất cả các hard alignment có thể xảy ra, mà sử dụng cơ chế attention để trực tiếp tính toán soft alignment giữa dữ liệu đầu vào và output label.

1.4 Data Augmentation

Khi dữ liệu dùng để huấn luyện mô hình bị hạn chế, giải pháp phù hợp nhất là thu thập thêm dữ liệu từ các nguồn tương tự. Nếu dữ liệu là một ngôn ngữ không phổ biến, việc thu thập thêm dữ liệu có thể gặp nhiều khó khăn. Một giải pháp khác là mô phỏng dữ liệu, việc tăng cường dữ liệu có thể giúp mô hình có kết quả tốt hơn với các dữ liệu âm thanh đầu vào bị nhiễu hoặc hiếm thấy. Các phương pháp đã được sử dụng có thể kể đến như: thêm vọng, thay đổi tốc độ tiếng nói, tạo ra dữ liệu mới và thêm nhiễu.

- **Thêm vọng** : Việc giả lập lại tiếng âm thanh dội lại từ các bề mặt được thực hiện bởi RIRs (Room Impulse Responses) vì tạo ra tiếng dội trên dữ liệu cho trước ở ngoài đời sẽ rất khó khăn.
- **Thay đổi tốc độ nói** : Phương pháp này kéo dài hoặc nén lại thời lượng của tín hiệu âm thanh đầu vào bằng cách thay đổi sample rate của dạng sóng của âm thanh.
- **Tạo ra dữ liệu mới**: Ta có thể sử dụng mô hình TTS (Text-to-Speech) huấn luyện để trả về giọng nói giống với trong dữ liệu, tạo ra các âm thanh mới với giọng điệu khác cho bộ dữ liệu.
- **Thêm nhiễu**: Giải pháp phổ biến hiện nay cho việc thêm nhiễu là sử dụng SpecAugment. Phương pháp này sẽ xử lý trực tiếp trên phổ tín hiệu thay vì dạng sóng của âm thanh như trong phương pháp thay đổi tốc độ nói, thực hiện dùng mặt nạ để che một số vùng trong phổ tín hiệu. Từ đó có thể tái tạo lại các trường hợp người nói nói sai hoặc có nhiễu trong máy thu âm. SpecAugment tăng cường dữ liệu rất hiệu quả cho các mô hình End-to-End.



Hình 3: SpecAugmentation

2 ASR Data Agumentation

2.1 Tập dữ liệu LibriSpeech

LibriSpeech là một nguồn dữ liệu quan trọng được xây dựng từ các bản ghi âm đọc sách, chủ yếu được lấy từ dự án LibriVox, nơi cộng đồng người đọc tự do cung cấp các bản đọc sách miễn phí. Tập dữ liệu chính này với tổng thời lượng thu âm vượt quá 1000 giờ, là một kho tài nguyên đa dạng và phong phú về ngôn ngữ và nội dung. Tuy nhiên với quy mô của đồ án lần này, nhóm sẽ huấn luyện mô hình trên tập train-clean-100.

Điều này để đảm bảo hiệu suất mô hình và tối ưu hóa quá trình huấn luyện. Tập dữ liệu train-clean-100 bao gồm 100 giờ ghi âm của các đoạn thoại đọc từ sách, với đặc điểm nổi bật là chỉ chứa âm thanh không bị nhiễu, hay còn được gọi là bản clean. Điều này phù hợp với tiêu chí xây dựng mô hình lần này, vì nó giúp giảm ảnh hưởng của nhiễu và cải thiện kết quả mô hình. Ngoài ra nhóm sử dụng tập dev-clean để validation và đánh giá bằng tập test-clean.

2.2 Xử lý dữ liệu âm thanh

Đầu tiên nhóm sẽ chuyển định dạng các file âm thanh từ .flac sang .wav sử dụng thư viện sox:

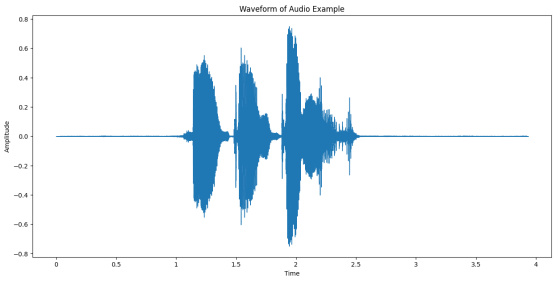
```
flac_list = glob.glob('/content/valid/LibriSpeech/dev-clean/**/**/*.flac',
    recursive=True)
for flac_path in flac_list:
    wav_path = flac_path[:-5] + '.wav'
    cmd = ["sox", flac_path, wav_path]
    subprocess.run(cmd)
```

sử dụng thư viện Ipython để nghe được file âm thanh trong jupyter notebook

```
import IPython.display as ipd
# Load and listen to the audio file
example_file = cvoice["train"][1]
audio_path = example_file["path"]
audio, sample_rate = librosa.load(audio_path)
ipd.Audio(audio, rate=sample_rate)
```

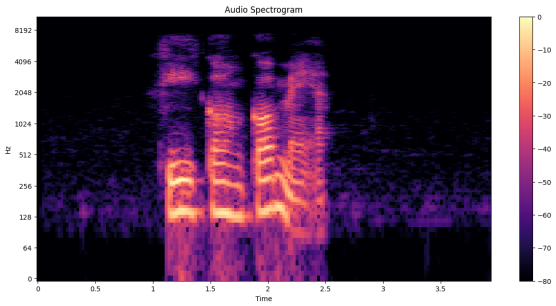
Biểu diễn âm thanh bằng đồ thị dạng sóng theo hai giá trị là biên độ và thời gian

```
%matplotlib inline
import librosa.display
import matplotlib.pyplot as plt

# Plot our example audio file's waveform
plt.rcParams['figure.figsize'] = (15,7)
plt.title('Waveform of Audio Example')
plt.ylabel('Amplitude')

_ = librosa.display.waveshow(audio, color='blue')
```



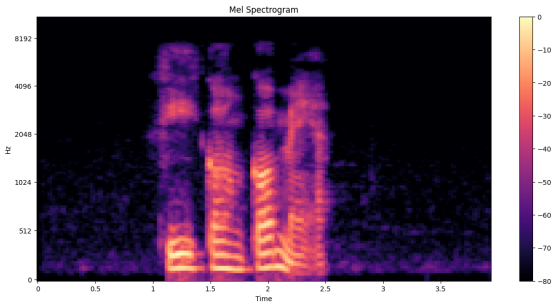
Dữ liệu Audio hiếm khi được đưa trực tiếp vào các Deep Learning model để huấn luyện. Thay vào đó, chúng thường được chuyển sang dạng Spectrogram sử dụng phép biến đổi Fourier, trong đó trục X thể hiện thời gian, trục Y thể hiện tần số và độ lớn của biên độ được thể hiện thông qua màu sắc. Màu sắc càng sáng thì thì biên độ càng lớn và ngược lại

```
import numpy as np

# Get spectrogram using Librosa's Short-Time Fourier Transform (stft)
spec = np.abs(librosa.stft(audio))
spec_db = librosa.amplitude_to_db(spec, ref=np.max) # Decibels

# Use log scale to view frequencies
librosa.display.specshow(spec_db, y_axis='log', x_axis='time')
plt.colorbar()
plt.title('Audio Spectrogram');
```



Khả năng nhận thức âm thanh của con người về hầu hết những âm thanh mà chúng nhóm nghe được đều tập trung xung quanh một dải tần số và biên độ khá hẹp. Do đó nhóm sẽ đưa về biểu đồ Mel Spectrogram, scale lại tần số từ hệ tuyến tính về hệ mel tức là thay thế giá trị tần số hiện tại bằng giá trị logarit của nó

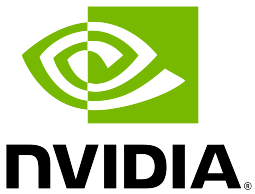
```
# Plot the mel spectrogram of our sample
mel_spec = librosa.feature.melspectrogram(y=audio, sr=sample_rate)
mel_spec_db = librosa.power_to_db(mel_spec, ref=np.max)

librosa.display.specshow(
    mel_spec_db, x_axis='time', y_axis='mel')
plt.colorbar()
plt.title('Mel Spectrogram');
```



2.3 NeMo toolkit cho ASR

NVIDIA NeMo, được tích hợp trong nền tảng NVIDIA AI, là một bộ công cụ phát triển các mô hình AI hội thoại tiên tiến. NeMo có các bộ mô hình riêng biệt cho các mô hình nhận dạng giọng nói tự động (ASR), xử lý ngôn ngữ tự nhiên (NLP) và tổng hợp giọng nói từ văn bản (TTS). Mỗi bộ mô hình bao gồm các mô đun được xây dựng sẵn, cung cấp mọi thứ cần thiết để huấn luyện trên dữ liệu của người dùng. Mỗi mô đun đều có thể dễ dàng tùy chỉnh, mở rộng và kết hợp để tạo ra các kiến trúc mô hình AI hội thoại mới.

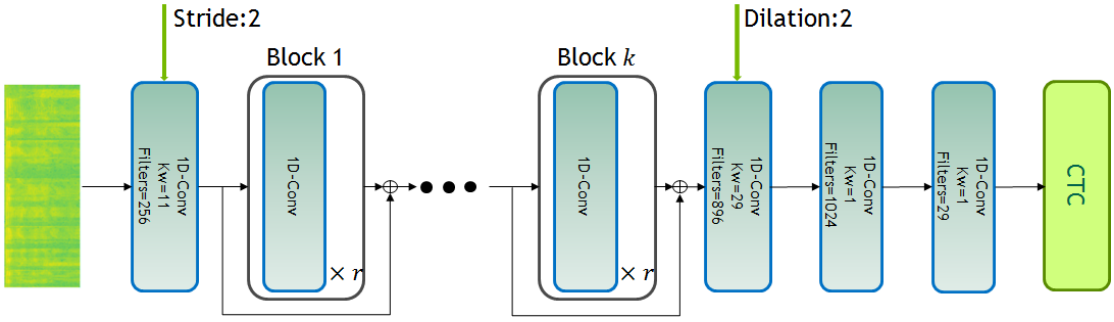
Kiến trúc hội thoại thường có quy mô lớn và đòi hỏi nhiều dữ liệu để tính toán và huấn luyện do đó NeMo sử dụng PyTorch Lightning để hun luyện multi-GPU / multi-node training dễ dàng và hiệu quả.



2.4 Mô hình ASR

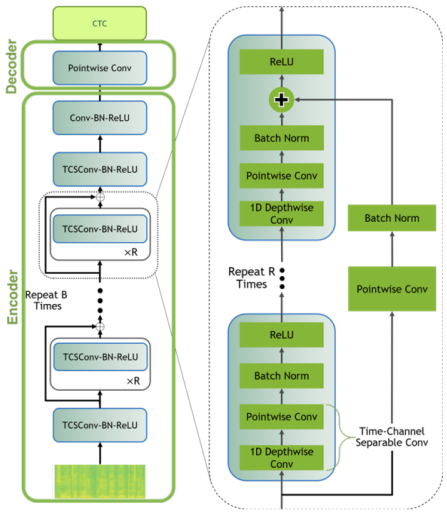
2.4.1 Mô hình Jasper

Nhóm sẽ huấn luyện một mô hình Jasper (Just Another SPeech Recognizer) nhỏ từ đầu . Về cơ bản, kiến trúc Jasper bao gồm một cấu trúc khối lặp lại sử dụng các phép toán tích chập một chiều (1D convolutions). Trong mô hình Jasper_KxR, tức là có R khối phụ (gồm một phép toán tích chập một chiều, chuẩn hóa theo lô, ReLU và dropout) được nhóm thành một khối đơn sau đó được lặp lại K lần. Đồng thời cũng có một khối bổ sung ở đầu và một vài khối khác ở cuối không phụ thuộc vào K và R sử dụng CTC loss function.



2.4.2 Mô hình QuartzNet

QuartzNet là một biến thể tốt hơn của Jasper với điểm khác biệt chính là nó sử dụng các phép toán tích chập một chiều tách biệt time-channel. Điều này cho phép giảm đáng kể số lượng trọng số trong khi vẫn duy trì độ chính xác tương tự.

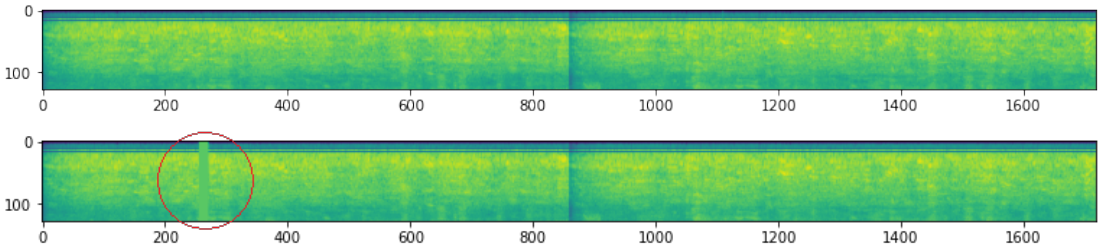


2.5 Data Agumentation

Bộ công cụ Nemo hỗ trợ sử dụng SpecAugment lên mô hình, nhóm sẽ thực hiện hai phương pháp trong SpecAugment để củng cố tập dữ liệu là Time Masking và Frequency Masking:

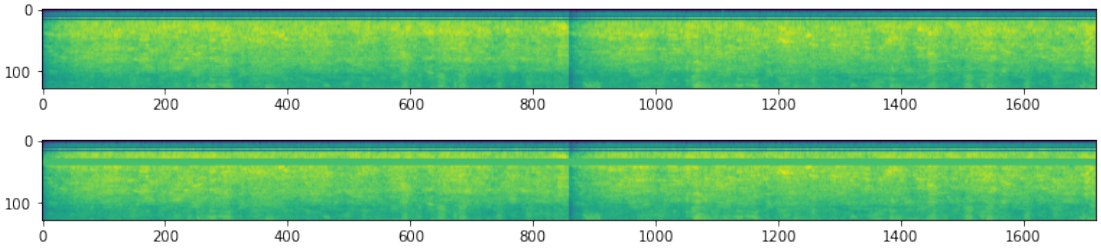
2.5.1 Time Masking

Phương pháp time masking mask t bước thời gian liên tiếp, nằm trong khoảng $[t_0, t_0 + t)$, trong đó t được chọn từ một phân phối đồng đều từ 0 đến tham số time mask T, và t_0 được chọn từ khoảng $[0, \tau - t)$ trong đó τ là độ dài của file âm thanh.



2.5.2 Frequency masking

Tương tự như time masking, phương pháp frequency masking mask f kênh mel frequency liên tiếp, nằm trong khoảng $[f_0, f_0 + f)$. Trong đó f được chọn từ một phân phối đồng đều từ 0 đến tham số frequency mask F, và f_0 được chọn từ khoảng $[0, \nu - f)$ trong đó ν là tổng số kênh mel frequency.



2.5.3 Sử dụng

Nhóm áp dụng Time Masking và Frequency Masking vào mô hình thông qua các tham số trong tệp cấu hình của Nemo ASR. Đối với Time Masking, giá trị **time_masks** sẽ chọn ra 5 time band ngẫu nhiên để áp dụng mask và giá trị **time_width** chính là độ rộng tối đa của mỗi time band đó. Tương tự, Frequency Masking cũng sử dụng tham số **freq_masks** chọn ra 5 frequency band ngẫu nhiên và **freq_width** để xác định đồ rộng tối đa.

```
model:
 spec_augment:
    _target_: nemo.collections.asr.modules.SpectrogramAugmentation
```

```
    # SpecAugment parameters
    freq_masks: 2 # Cut two frequency bands
    freq_width: 15 # ... of width 15 at maximum
    time_masks: 5 # Cut out 5 time bands
    time_width: 25 # ... of width 25 at maximum
```

2.6 Thực hiện và kết quả

2.6.1 Thực hiện

Trong Nemo ASR, nội dung của tệp manifest tương thích có dạng:

```
    {"audio_filepath": "path/to/audio.wav",
    "duration": 3.45,
    "text": "this is a nemo tutorial"}
```

Xây dựng hai tệp train_manifest và valid_manifest theo trên định dạng trên dựa vào thông tin được cung cấp trong file "trans.txt" của tập dữ lựu LibriSpeech.

```
import glob
import librosa

manifest = glob.glob('/content/train/LibriSpeech/dev-clean/**/**/*.trans.txt',
    recursive=True)
with open(train_manifest, 'w') as fout:
  for batch in manifest:
    with open(batch, 'r') as fin:
      for line in fin:
        # Splitting the line into four parts
        parts = line.split(' ', 2)

        # Extracting the first three parts
        path1, path2, path3 = parts[0].split('-')

        # Joining the remaining parts to get the fourth part
        transcript = ' '.join(parts[1:]).strip()

        audio_path = "/content/train/LibriSpeech/dev-clean/" + path1 + '/' + path2 + '/'
            + path1 + '-' + path2 + '-' + path3 +'.wav'
        duration = librosa.core.get_duration(filename=audio_path)

        # Write the metadata to the manifest
        metadata = {
            "audio_filepath": audio_path,
            "duration": duration,
            "text": transcript
        }
        json.dump(metadata, fout)
        fout.write('\n')
```

Nhóm sẽ sử dụng tệp cấu hình config.yaml được gợi ý từ Nemo github chuyên dùng trong các bài toán ASR và Experiment Manager để quản lý việc huấn luyện và lưu các checkpoint. Cấu hình cho exp_manager như sau:

```
exp_manager:
  exp_dir: /content/gdrive/MyDrive/NeMoLib
  name: *name
  version: version_1
  use_datetime_version: False
  create_tensorboard_logger: True
  create_checkpoint_callback: True
  resume_if_exists: True
  resume_ignore_no_checkpoint: True
  checkpoint_callback_params:
    save_top_k: 1 # Dont save too many .ckpt files during HP search
    always_save_nemo: True
```

Nạp các tham số từ file config vào trainer và exp_manager để bắt đầu train mô hình, nhóm sẽ train hai mô hình trong đó có một mô hình không sử dụng data augmentation để so sánh kết quả:

```
import nemo
import nemo.collections.asr as nemo_asr
import pytorch_lightning as pl
from omegaconf import DictConfig
from nemo.utils.exp_manager import exp_manager

trainer = pl.Trainer(max_epochs=40, devices=params['trainer']['devices'],
    accelerator=params['trainer']['accelerator'], logger=params['trainer']['logger'],
    enable_checkpointing=params['trainer']['enable_checkpointing'])

exp_manager(trainer, cfg=DictConfig(params['exp_manager']))

params['model']['train_ds']['manifest_filepath'] = train_manifest
params['model']['validation_ds']['manifest_filepath'] = valid_manifest
first_asr_model = nemo_asr.models.EncDecCTCModel(cfg=DictConfig(params['model']),
    trainer=trainer)
# Start training!!!
trainer.fit(first_asr_model)
```

2.7 Kết quả

Sử dụng tập test-clean để đánh giá kết quả sau khi train xong hai mô hình. Nhóm sử dụng đoạn code sau để trả về giá trị WER.

```
wer_nums = []
wer_denoms = []

# Loop over all test batches.
# Iterating over the model's `test_dataloader` will give us:
# (audio_signal, audio_signal_length, transcript_tokens, transcript_length)
# See the AudioToCharDataset for more details.
```

```
for test_batch in asr_model.test_dataloader():
        test_batch = [x.cuda() for x in test_batch]
        targets = test_batch[2]
        targets_lengths = test_batch[3]
        log_probs, encoded_len, greedy_predictions = asr_model(
            input_signal=test_batch[0], input_signal_length=test_batch[1]
        )
        # Notice the model has a helper object to compute WER
        asr_model._wer.update(greedy_predictions, targets, targets_lengths)
        _, wer_num, wer_denom = asr_model._wer.compute()
        asr_model._wer.reset()
        wer_nums.append(wer_num.detach().cpu().numpy())
        wer_denoms.append(wer_denom.detach().cpu().numpy())

        # Release tensors from GPU memory
        del test_batch, log_probs, targets, targets_lengths, encoded_len,
            greedy_predictions

# We need to sum all numerators and denominators first. Then divide.
print(f"WER = {sum(wer_nums)/sum(wer_denoms)}")
```

	Batch size	**Epoch**	**Thời gian**	**WER**
QuartzNet with data agumentation	32	40	20h	0.43929
QuartzNet without data agumentation	32	40	20h	0.51221

Có thể thấy rằng đối với mô hình có áp dụng data agumention có kết quả tốt hơn so với mô hình gốc. Qua đó cho thấy vai trò của việc data agumention trong việc cải thiện mô hình ASR hiệu quả hơn với những dữ liệu âm thanh trong đa dạng môi trường và hoàn cảnh.

3 Kết luận

Đối với cả hai mô hình được train, kết quả cuối cùng chưa phải là tốt nhất do thời gian train chưa đủ lâu. Đồng thời trong đồ án lần này chỉ sử dụng các cấu hình cơ bản nhất trong môi trường NeMo Toolkit để huấn luyện một mô hình ASR và tập data sử dụng cũng chưa đủ lớn để mô hình học hết các từ trong từ điển. Ngoài ra việc chỉ sử dụng tập data clean khiến cho không thể đánh giá cũng như cải thiện mô hình trong các môi trường phức tạp và đa dạng khác. Em xin cảm ơn thầy đã hướng dẫn và giúp đỡ hoàn thiện đồ án, tuy còn nhiều hạn chế do còn chưa làm quen nhiều với huấn luyện mô hình nhưng bản thân học được nhiều kinh nghiệm và kiến thức quan trọng.

References

[1] Pranav Putta, *Solving N Queens for 1 Million Queens with MinConflict*, https://medium.com/@pranav.putta22/solving-n-queens-for-1-million-queens-with-minconflict-62ef798556e0

[2] Tomáš Müller, *Interactive Heuristic Search Algorithm*, https://muller.unitime.org/ihsa02.pdf

[3] HRoman Barták, *Heuristics and Stochastic Algorithms*, https://ktiml.mff.cuni.cz/~bartak/constraints/stochastic.html

[4] *Nemo ASR tutorial*, https://github.com/NVIDIA/NeMo/blob/main/tutorials/asr/ASR_with_NeMo.ipynb

[5] zcaceres, *SpecAugment with Pytorch*, https://github.com/zcaceres/spec_augment

[6] OpenSLR, *LibriSpeech ASR corpus*, https://www.openslr.org/12

[7] Samuel Kriman, Stanislav Beliaev, Boris Ginsburg, Jocelyn Huang, Oleksii Kuchaiev, Vitaly Lavrukhin, Ryan Leary, Jason Li, Yang Zhang, *QuartzNet: Deep Automatic Speech Recognition with 1D Time-Channel Separable Convolutions*, https://arxiv.org/abs/1910.10261