BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CẦN THƠ
TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

NIÊN LUẬN
NGÀNH KHOA HỌC MÁY TÍNH

Đề tài

XÂY DỰNG ỨNG DỤNG TẠO GIỌNG NÓI BẰNG DEEP LEARNING

Sinh viên thực hiện : Trần Anh Khoa
Mã số : B1913240
Khóa : 45

Cần Thơ, 11/2022

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CẦN THƠ

TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

NIÊN LUẬN

NGÀNH KHOA HỌC MÁY TÍNH

Đề tài

XÂY DỰNG ỨNG DỤNG TẠO GIỌNG NÓI BẰNG DEEP LEARNING

Giáo viên hướng dẫn:
Th.S.Phạm Nguyên Hoàng

Sinh viên thực hiện: Trần Anh Khoa
Mã số: B1913240
Khoá: 45

Cần Thơ, 11/2022

NHẬN XÉT CỦA GIẢNG VIÊN

LỜI CẢM ƠN

Lời đầu tiên, em xin chân thành cảm ơn thầy Phạm Nguyên Hoàng, thầy đã tận tình hướng dẫn, cung cấp những kiến thức quý báu cho em trong suốt quá trình thực hiện niên luận chuyên ngành. Cảm ơn thầy đã tạo điều kiện và cho em cơ hội để thực hiện đề tài này. Cảm ơn sự tâm huyết của thầy trong quá trình trình giảng dạy, ngọn lửa nhiệt huyết của thầy truyền lại sẽ là một hành trang cho em vững bước trên con đường mưu cầu tri thức.

Mặc dù đã cố gắng và nỗ lực rất nhiều trong quá trình thực hiện đề tài này. Tuy nhiên, do kiến thức và kĩ năng còn hạn chế nên khó tránh những sai sót. Em rất mong được sự thông cảm, góp ý của thầy để niên luận chuyên ngành của em được hoàn thiện hơn.

Cần Thơ, ngày 14 tháng 11 năm 2022

Người viết

Trần Anh Khoa

MỤC LỤC

DANH MỤC HÌNH

ABSTRACT

With the development of modern technology today, voice cloning has become a development trend for many companies around the world. Voice cloning helps a lot in everyday life such as creating artificial voices for patients who cannot talk without assistance, such as those who have lost their voice after surgery or illness. Voice cloning can also be used to translate an actor's speech into different languages, thus meaning production companies will no longer need to hire additional actors to do dub versions of sound for their films for overseas distribution.

The topic "Building a voice cloning application using deep learning" will contribute significantly to the development in many areas of our lives in the future.

TÓM TẮT

Với sự phát triển của công nghệ hiện đại ngày nay, nhân bản giọng nói đã và đang trở thành xu hướng phát triển của nhiều công ty trên thế giới. Nhân bản giọng nói giúp ích rất nhiều trong cuộc sống hàng ngày chẳng hạn như tạo ra giọng nói nhân tạo cho những bệnh nhân không thể nói chuyện mà không có sự trợ giúp, chẳng hạn như những người bị mất giọng nói sau phẫu thuật hoặc bệnh tật. Nhân bản giọng nói cũng có thể được sử dụng để dịch lời nói của một diễn viên sang các ngôn ngữ khác nhau, do đó có nghĩa là các công ty sản xuất phim sẽ không còn cần phải thuê thêm diễn viên để làm phiên bản lồng tiếng cho phim của họ để phân phối ở nước ngoài.

Đề tài "Xây dựng ứng dụng nhân bản giọng nói bằng deep learning" sẽ góp phần phát triển không nhỏ ở nhiều lĩnh vực trong cuộc sống của chúng ta trong tương lai.

PHẦN GIỚI THIỆU

1. Đặt vấn đề

Trong bối cảnh thời đại công nghiệp hoá, hiện đại hoá như ngày nay, nhân bản giọng nói là một ngành khoa học mới nổi và là một chủ đề gây tranh cãi. Một số người nghĩ rằng nó có thể cách mạng hóa giao tiếp của con người, trong khi những người khác tin rằng đó là thứ khoa học viễn tưởng. Về mặt kỹ thuật được gọi là sao chép giọng nói hoặc tổng hợp giọng nói, nhân bản giọng nói là quá trình tạo ra một giọng nói mới với các đặc điểm giống với giọng nói hiện có. Một cách để tạo ra một giọng nói mới bằng cách sử dụng học sâu là thông qua tổng hợp dạng sóng. Dưới đây là một số lợi ích của việc sử dụng nhân bản giọng nói trong cuộc sống hàng ngày:

Học sâu được sử dụng để nhân bản giọng nói của con người. Đó là một kỹ thuật AI phức tạp hơn nhiều so với các phương pháp trước đây được sử dụng để sửa đổi giọng nói. Mục tiêu chính của nhân bản giọng nói là tạo ra một giọng nói giống với giọng nói tự nhiên của con người mà không sử dụng bất kỳ giọng nói riêng lẻ nào của con người. Nhiều kỹ thuật khác nhau đã được phát triển cho mục đích này, bao gồm mạng nơron học sâu, tổng hợp dạng sóng và điều chế nhiễu. Về lý thuyết, tất cả các giọng nói đều có thể được tạo ra thông qua tổng hợp giọng nói nhân bản. Tuy nhiên, chất lượng của giọng nói tổng hợp giảm khi số lượng giọng nói tăng lên. Do đó, cho đến nay chỉ có một số giọng nói tổng hợp có âm thanh tự nhiên tồn tại. Tuy nhiên, công nghệ này vẫn còn trong giai đoạn sơ khai và chắc chắn sẽ tiến bộ khi có nhiều người làm việc trên nó .

Do nhận thấy được tầm quan trọng của giọng nói nên vào năm 2014, giáo sư Patel thành lập doanh nghiệp tạo ra giọng nói nhân tạo cho những bệnh nhân không thể nói chuyện mà không có sự trợ giúp, chẳng hạn như những người bị mất giọng nói sau phẫu thuật hoặc bệnh tật. Patel cho biết công nghệ - được dẫn dắt bởi trí tuệ nhân tạo (AI), phần mềm có thể tự "học" và thích ứng - đã phát triển vượt bậc trong vài năm qua. Điều này đã thu hút sự chú ý từ giới nghệ sĩ lồng tiếng. Nhân bản giọng nói cũng có thể được sử dụng để dịch lời nói của một diễn viên sang các ngôn ngữ khác nhau, do đó có nghĩa là các công ty sản xuất phim của Mỹ sẽ không còn cần phải thuê thêm diễn viên để làm phiên bản lồng tiếng cho phim của họ để phân phối ở nước ngoài.

Tuy nhiên, mặc dù sự tinh vi ngày càng tăng của nhân bản giọng nói có tiềm năng thương mại rõ ràng, điều đó cũng dẫn đến mối lo ngại ngày càng tăng rằng công nghệ có thể được sử dụng trong thế giới tội phạm mạng để lừa đảo. Cùng với các video giả do máy tính tạo ra, nhân bản giọng nói còn được gọi là "deepfake".

Chuyên gia an ninh mạng Eddy Bobritsky nhận định có một "nguy cơ bảo mật rất lớn" đi kèm với những tiếng nói tổng hợp. Bobritsky, ông chủ Công ty Minerva Labs của Israel, bình luận: "Khi nói đến email hoặc tin nhắn văn bản, chúng ta biết rằng việc mạo danh người khác trong nhiều năm là khá dễ dàng. Nhưng, cho đến nay, nói chuyện qua điện thoại với người mà bạn tin tưởng và biết rõ là một trong những cách phổ biến nhất để đảm bảo rằng bạn thực sự quen thuộc với người đó". Nhưng Bobritsky tuyên bố điều đó hiện đang thay đổi: "Ví dụ, nếu sếp gọi điện cho một nhân viên yêu cầu cung cấp thông tin nhạy cảm và nhân viên nhận ra giọng nói, phản ứng ngay lập tức là làm theo yêu cầu. Đó là lợi thế rất lớn cho nhiều tội phạm mạng".

Nhân bản giọng nói là một ngành khoa học mới nổi có thể cách mạng hóa giao tiếp của con người- hoặc đưa chúng ta đi thẳng vào lãnh thổ khoa học viễn tưởng nếu không được kiểm soát. Học sâu đã được chứng minh là hữu ích trong việc phát triển lĩnh vực này cho đến nay nhưng có thể sẽ không đủ khi cố gắng tái tạo giọng nói thực tế của con người ở dạng tổng hợp. Sự thật của sự tổng hợp giọng nói nhân bản vẫn chưa được biết. Tuy nhiên, không thể phủ nhận những khả năng trong tương lai khi chúng ta tiếp tục phát triển công nghệ mới với tốc độ chóng mặt!

2. Lịch sử giải quyết vấn đề

Vào thời điểm trợ lý giọng nói lần đầu tiên được giới thiệu trên điện thoại thông minh. Mặc dù khả năng của trợ lý giọng nói lúc đó rất hạn chế. Nhưng, khái niệm nói chuyện với điện thoại khiến mọi người cảm thấy hào hứng. Một trong những hạn chế của trợ lý giọng nói ban đầu là chúng phát ra âm thanh rất không tự nhiên và máy móc. Tuy nhiên, theo thời gian cả khả năng nói và chức năng của những trợ lý này cũng đã phát triển. Bây giờ, trọng âm, giọng điệu, cao độ, v.v. của các tác nhân giọng nói nghe rất tự nhiên. Tất cả đã trở nên khả thi nhờ những tiến bộ trong lĩnh vực trí tuệ nhân tạo và chuyển văn bản thành giọng nói trong những năm qua.

Hình 1. Minh hoạ trợ lý ảo giọng nói

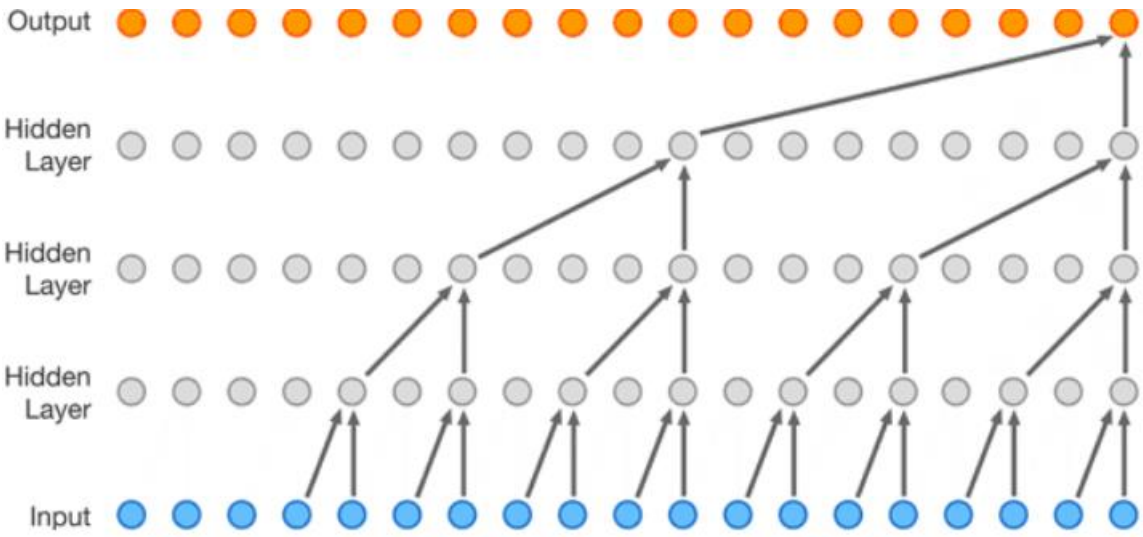
Giờ đây, các mạng nơ-ron có thể chỉ mất một vài giây để phân tích giọng nói của bạn và tạo ra các mẫu âm thanh hoàn toàn tự nhiên mới. Hơn nữa, những giọng nói tổng hợp này có thể sớm không thể phân biệt được với các mẫu âm giọng nói gốc. Nếu bạn cố gắng so sánh một số ví dụ về sao chép giọng nói, bạn sẽ dễ dàng đánh giá được phạm vi của những gì công nghệ có thể làm, bao gồm khả năng chuyển đổi giới tính của giọng nói, thay đổi trọng âm và phong cách nói.

Chúng ta đang ngày càng tiến tới một thế giới điều khiển bằng giọng nói. Việc tiêu thụ những nội dung liên quan âm thanh và các dịch vụ tự động dựa trên giọng nói đang gia tăng. Nhiều người sáng tạo nội dung đang chuyển sang các nền tảng như SoundCloud và dịch vụ sách nói của Amazon, Audible. Cũng có thể cảm nhận được từ thực tế là những gã khổng lồ công nghệ như Google, Amazon, Samsung, Apple, v.v., đang đầu tư rất nhiều vào các dịch vụ dựa trên giọng nói của họ và họ thường tuyên bố là tốt hơn các đối tác của mình.

Chúng ta sẽ điểm qua một số bước phát triển đột phá trong lĩnh vực nhân bản giọng nói trong những năm gần đây.

Đầu tiên là công nghệ WaveNet, nó được giới thiệu vào năm 2016 bởi Google-backed DeepMind trong bài báo về WaveNet: Mô hình sáng tạo cho âm thanh thô. Các hệ thống chuyển văn bản thành giọng nói (TTS) trước đó phần lớn dựa trên TTS nối. Trong cách tiếp cận này, đầu tiên, một cơ sở dữ liệu rất lớn gồm các đoạn thoại ngắn được ghi lại từ một người nói. Sau đó, các đoạn này được kết hợp lại để tạo thành các phát biểu hoàn chỉnh. Nhược điểm của phương pháp này là bạn sẽ cần một cơ sở dữ liệu mẫu âm thanh hoàn toàn mới nếu bạn muốn thực hiện các chỉnh sửa nhỏ đối với giọng nói, chẳng hạn như thay đổi điểm nhấn hoặc cảm xúc. Ngoài ra,

các mẫu âm thanh được tạo ra bởi phương pháp này rất không tự nhiên, rối mắt và giống như robot. Mặt khác, WaveNet là một TTS tham số, trong đó tất cả thông tin cần thiết để tạo dữ liệu được lưu trữ trong các tham số của mô hình. Bạn có thể kiểm soát các đặc điểm của giọng nói thông qua các đầu vào của mô hình. WaveNet tạo đầu ra của nó bằng cách lập mô hình trực tiếp dạng sóng thô của tín hiệu âm thanh, mỗi lần một mẫu. Các mẫu âm thanh do WaveNet tạo ra nghe tự nhiên hơn so với TTS nối. Chúng ta có thể xem các mẫu bài phát biểu trên blog của DeepMind để hiểu sự khác biệt về tính tự nhiên của bài phát biểu được tổng hợp bởi các phương pháp này.



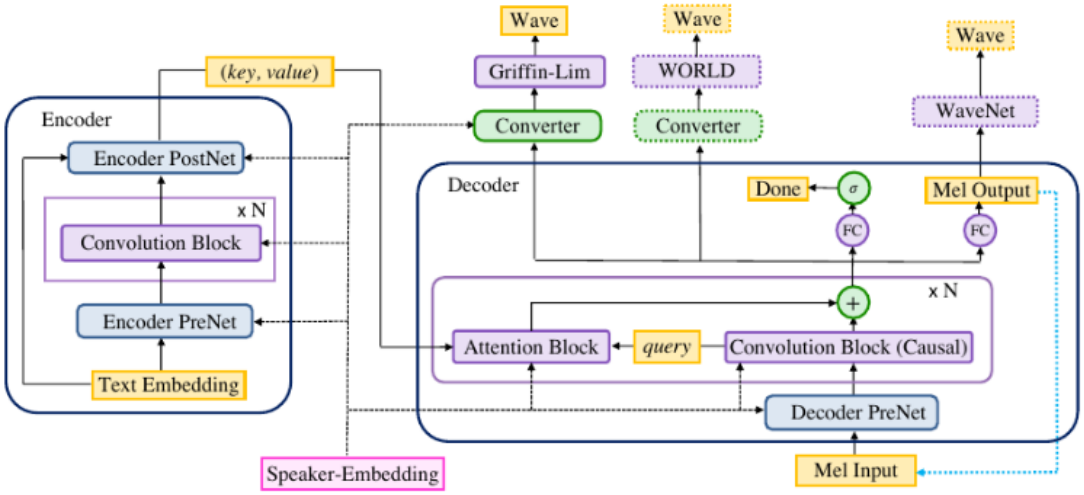
Hình 2. Các khối xây dựng của Mô hình học sâu WaveNet

Thành phần chính của WaveNet là tập hợp các cấu trúc thông thường. Ở đây, các lớp tích chập có nhiều hệ số giãn nở khác nhau cho phép trường tiếp nhận của nó phát triển theo cấp số nhân với độ sâu và bao phủ hàng nghìn bước thời gian. Về cơ bản, trong một lớp tích chập giãn nở, bộ lọc được áp dụng trên một diện tích lớn hơn chiều dài của nó bằng cách bỏ qua các giá trị đầu vào với một bước nhất định. Điều này tương tự với gộp hoặc tích chập theo từng bước, nhưng ở đây, đầu ra có cùng kích thước với đầu vào. Vì vậy, không có lớp tổng hợp nào trong mạng và đầu ra của mô hình có cùng chiều thời gian với đầu vào. Bằng cách sửa đổi danh tính người nói, WaveNet có thể được sử dụng để nói điều tương tự bằng các giọng nói khác nhau. Tương tự, các đầu vào bổ sung như cảm xúc hoặc trọng âm có thể được cung cấp cho mô hình để làm cho bài phát biểu trở nên đa dạng và thú vị hơn. Nó cũng có thể bắt chước những âm thanh không lời, chẳng hạn như hơi thở và chuyển động của miệng. Vì WaveNets có thể được sử dụng để mô hình hóa bất kỳ tín hiệu âm thanh nào nên nó cũng có thể được sử dụng để tạo nhạc! Một nhược điểm của các mô hình tự hồi

quy như WaveNet là chúng có xu hướng học cấu trúc cục bộ tốt hơn nhiều so với cấu trúc toàn cục. Nó đáng chú ý hơn khi lập mô hình phân phối chiều cao. Do đó dẫn đến sự ra đời của MelNet là một mô hình tổng quát giải quyết vấn đề này bằng cách sử dụng mô hình đa tỷ lệ. Theo bài báo, MelNet có thể nắm bắt rất tốt cấu trúc cấp cao của âm thanh.

Kế tiếp là mô hình **Deep Voice**, là một hệ thống chuyển văn bản thành giọng nói (**TTS**) được thành lập bởi các nhà nghiên cứu tại Baidu. Đây là phiên bản đầu tiên, **Deep Voice 1** được lấy cảm hứng bởi các đường dẫn chuyển văn bản thành giọng nói truyền thống. Nó thông qua cấu trúc tương tự, nhưng thay thế tất cả các thành phần bằng mạng nơ ron và sử dụng các tính năng đơn giản hơn. Đầu tiên, nó chuyển các chữ thành các âm vị và sau đó sử dụng mô hình tổng hợp âm thanh để chuyển đổi các đặc điểm ngôn ngữ thành lời nói.

Phiên bản mới nhất là **Deep Voice 3**, trong đó sử dụng một kiến trúc từ ký tự sang quang phổ tích chập hoàn toàn. Kiến trúc này cho phép tính toán song song hoàn toàn, đào tạo nhanh hơn các mạng hồi quy. Kiến trúc được lấy cảm hứng từ máy biến áp (Vaswani et al). **Deep Voice 3** là hệ thống **TTS** đầu tiên mở rộng tới hàng nghìn người nói mới một mô hình duy nhất.



Hình 3. Minh hoạ kiến trúc Deep Voice 3

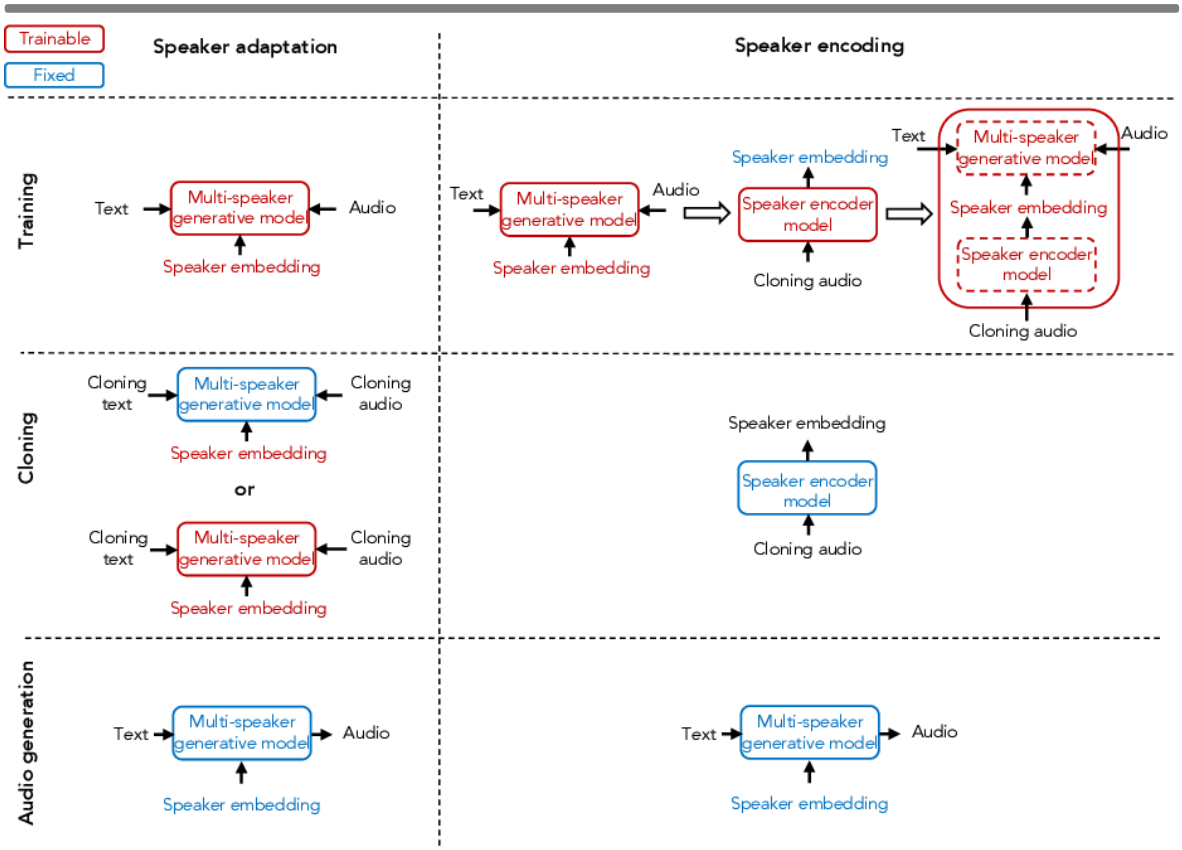
Kiến trúc của **Deep Voice 3** bao gồm ba thành phần:

- **Bộ mã hóa (Encoder):** Đây là một mạng thần kinh tích chập giúp chuyển đổi các đặc điểm văn bản thành một biểu diễn đã học bên trong.
- **Bộ giải mã (Decoder):** Bộ giải mã được sử dụng để giải mã biểu diễn đã học từ bộ mã hóa thành biểu diễn âm thanh có chiều thấp (hoặc phổ Mel). Nó bao gồm các kết hợp thông thường với sự chú ý tích chập multihop và nó tạo ra đầu ra của nó theo cách tự hồi quy.
- **Bộ chuyển đổi (Converter):** Nó là một mạng xử lý hậu tích chập hoàn toàn. Bộ chuyển đổi dự đoán các tham số cuối cùng từ các trạng thái ẩn của bộ giải mã. Điều này là phi nhân quả và do đó có thể phụ thuộc vào thông tin ngữ cảnh trong tương lai.

Hàm mục tiêu tổng thể được tối ưu hóa là sự kết hợp tuyến tính của các tổn thất từ bộ giải mã và bộ chuyển đổi. Các bộ phát âm sau đây có thể được sử dụng trong bộ chuyển đổi của Deep Voice 3: **Griffin-Lim vocoder**, **WORLD vocoder**, and **WaveNet vocoder**.

Vào cuối năm 2018, nhóm của Deep Voice đã phát hành bài báo: **Mạng Nơ-ron nhân bản giọng nói với một vài ví dụ**. Bài báo đã giới thiệu một hệ thống dựa trên Deep Voice 3 có nhiệm vụ nhân bản giọng nói. Hệ thống này có thể sao chép giọng nói chỉ trong vài giây với số lượng ví dụ rất ít. Thực tế, thời gian trung bình để sao chép là 3,7 giây!

Hai cách tiếp cận đã được các nhà nghiên cứu của Baidu áp dụng: **Thích ứng người nói (Speaker Adaptation)** và **Mã hóa người nói (Speaker Encoding)**. Cả hai phương pháp đều có thể mang lại hiệu suất tốt với dữ liệu đầu vào âm thanh tối thiểu và cả hai phương pháp này đều có thể được tích hợp vào hệ thống giọng trầm mà không làm giảm chất lượng của hệ thống.



Hình 4. Minh hoạ sự khác nhau giữa hai mô hình Thích ứng người nói và Mã hoá người nói

- **Thích ứng với người nói (Speaker Adaptation):** Mục tiêu của phương pháp này là tinh chỉnh mô hình nhiều người nói đã được đào tạo cho một người nói không nhìn thấy bằng cách sử dụng một vài cặp văn bản âm thanh. Có thể áp dụng tinh chỉnh cho phần nhúng người nói hoặc toàn bộ mô hình.

- **Mã hóa người nói (Speaker Encoding):** Phương pháp mã hóa người nói ước tính trực tiếp việc nhúng người nói từ các mẫu âm thanh của người nói không nhìn thấy. Một mô hình như thế này không yêu cầu bất kỳ tinh chỉnh nào trong quá trình nhân bản giọng nói. Do đó, cùng một mô hình có thể được sử dụng cho tất cả các giọng nói không nhìn thấy được.

Công nghệ tiếp theo cũng chính là mục tiêu của đề tài này nghiên cứu chính là **Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis (SV2TTS)** mô tả một hệ thống dựa trên mạng thần kinh để tổng hợp văn bản thành giọng nói (TTS) có khả năng tạo âm thanh giọng nói bằng giọng nói của những người nói khác nhau. Điều nổi bật của hệ thống này là nó có thể hoạt động trong cài đặt học tập **zero-shot**. Vì vậy, nó có thể tạo giọng nói mới bằng giọng nói của một người nói chưa từng thấy trước đó, chỉ sử dụng một vài giây âm thanh tham chiếu chưa được phiên âm mà không cần cập nhật bất kỳ tham số mô hình nào.

3. Mục tiêu đề tài

- Mục tiêu của đề tài là xây dựng ứng dụng nhân bản giọng nói sử dụng công nghệ deep learning với đầu vào là 1 đoạn audio giọng nói của một người bất kì dùng để làm mẫu. Sau đó sẽ tiến hành phân tích và tạo ra giọng nói giống với người dùng nhất có thể và dùng nó để đọc một đoạn văn bản bất kì.

4. Đối tượng và phạm vi nghiên cứu

✓ Hiện tại do khó khăn về mặt ngôn ngữ nên đối tượng và phạm vi nghiên cứu chỉ dừng lại ở ngôn ngữ Tiếng Anh.

5. Phương pháp nghiên cứu

Lý thuyết: Tham khảo các tài liệu được trích xuất từ các bài báo khoa học khác nhau về lĩnh vực SV2TTS.

Thực hành: Tìm hiểu cách thức hoạt động của những mô hình nhân bản giọng nói khác nhau và cài đặt mô hình trên máy.

6. Kết quả đạt được

- Cài đặt và triển khai thành công ứng dụng nhân bản giọng nói SV2TTS.

- Ứng dụng có thể tạo ra giọng nói nhân bản từ một đoạn audio giọng nói đầu vào chỉ vài giây và có thể dùng giọng nói đó đọc một đoạn văn bản bất kì.

7. Bố cục niên luận

Phần giới thiệu

Giới thiệu tổng quát về đề tài.

Phần nội dung

Chương 1 : Mô tả bài toán.

Chương 2 : Thiết kế, cài đặt giải thuật, biểu diễn cơ sở dữ liệu, trình bày các bước xây dựng hệ thống bằng phương pháp lọc cộng tác.

Chương 3 : Kiểm thử hệ thống và đánh giá độ chính xác, tốc độ của hệ thống.

Phần kết luận

Trình bày kết quả đạt được và hướng phát triển hệ thống.

PHẦN NỘI DUNG

CHƯƠNG 1

MÔ TẢ BÀI TOÁN

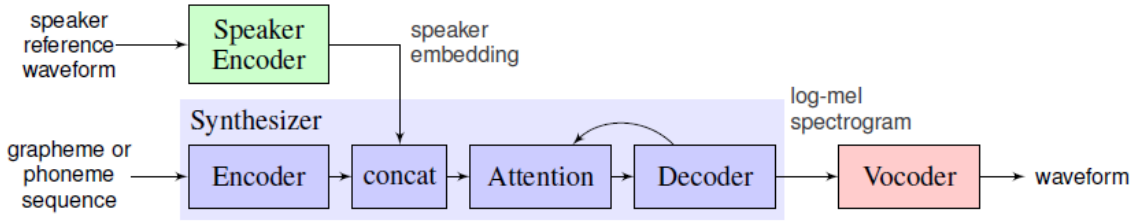
1. Mô tả chi tiết bài toán

Nhân bản giọng nói được hiểu là sử dụng một chương trình máy tính để tạo ra một bản sao tổng hợp, có thể điều chỉnh được từ giọng nói của một người. Từ bản ghi âm của ai đó đang nói chuyện, phần mềm có thể sao chép giọng nói của người đó và có thể đọc được một đoạn văn bản bất kì bằng giọng nói đã được sao chép.

2. Vấn đề và giải pháp liên quan đến bài toán

Các vấn đề chính và giải pháp liên quan để xây dựng ứng dụng nhân bản giọng nói bao gồm:

- Cách xử lý tín hiệu âm thanh
- Mô hình phân tích dữ liệu giọng nói (**Speaker Encoder** của mô hình SV2TTS)
- Mô hình SV2TTS (Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis)



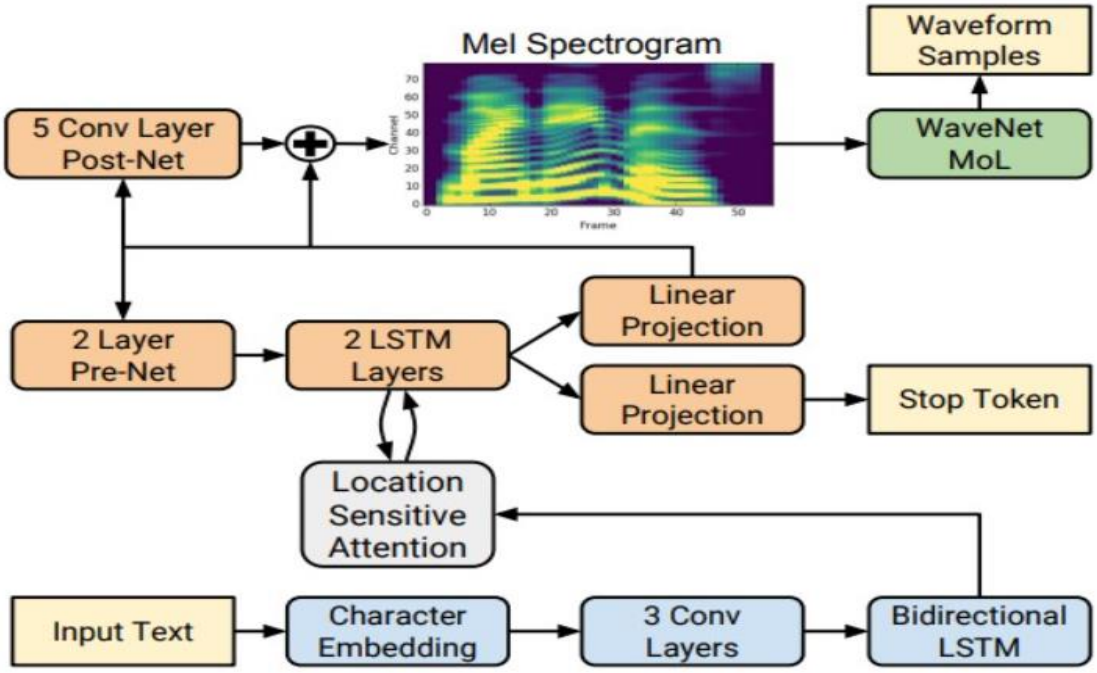
Hình 5. Ảnh minh hoạ kiến trúc SV2TTS

Hệ thống SV2TTS bao gồm ba thành phần được đào tạo độc lập. Điều này cho phép mỗi thành phần được đào tạo trên dữ liệu độc lập, giảm yêu cầu về dữ liệu nhiều người nói chất lượng cao. Các thành phần riêng lẻ bao gồm:

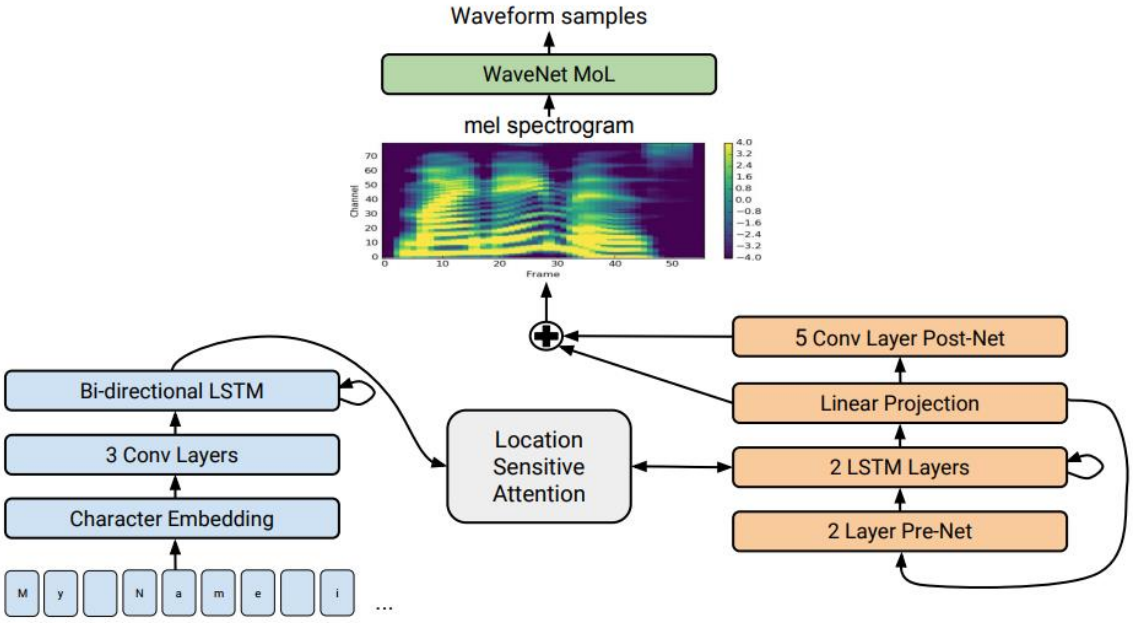
- **Mạng mã hoá giọng của người nói (Speaker Encoder):** Công việc của mạng mã hóa người nói là lấy âm thanh của một người nói nhất định làm đầu vào và mã hóa các đặc điểm giọng nói của họ thành một phép nhúng vectơ chiều thấp. Nó không quan tâm đến những gì người nói đang nói, tất cả những gì nó quan tâm là cách người nói nói điều gì đó. Mạng được đào tạo riêng về nhiệm vụ xác minh người nói, sử dụng bộ dữ liệu về lời nói có chứa tiếng ồn từ hàng nghìn người nói. Các mã hóa sau đó được sử dụng để tạo điều kiện cho mạng tổng hợp dựa trên tín hiệu giọng nói tham chiếu từ người nói mục tiêu mong muốn.

- **Mạng tổng hợp (Synthesizer):** Đây là mạng thần kinh **Seq2Seq** dựa trên **Tacotron 2** của google tạo ra quang phổ Mel từ văn bản, tùy thuộc vào việc nhúng giọng người nói. Đây là phiên bản mở rộng của **Tacotron 2** hỗ trợ nhiều người nói. Đầu ra được điều chỉnh theo giọng nói của người nói bằng cách ghép nối quá trình nhúng của chúng với đầu ra của bộ mã hóa tổng hợp tại mỗi bước thời gian.
- **Mạng phát âm (Vocoder):** Hệ thống sử dụng **WaveNet** làm bộ phát âm. Nó lấy các quang phổ Mel được tạo bởi mạng tổng hợp làm đầu vào và tự động tạo các dạng sóng âm thanh trong miền thời gian làm đầu ra. Mạng bộ tổng hợp được đào tạo sao cho nó cố gắng nắm bắt tất cả các chi tiết liên quan cần thiết để tổng hợp chất lượng cao của nhiều loại giọng nói dưới dạng phổ Mel. Điều này cho phép xây dựng bộ phát âm bằng cách huấn luyện đơn giản dựa trên dữ liệu từ nhiều người nói.

- Mô hình chuyển văn bản thành giọng nói (**Text To Speech**) ở đây chúng ta sử dụng kiến trúc **Tacotron 2**:



Hình 6. Ảnh minh hoạ kiến trúc Tacotron 2 (1)



Hình 7. Ảnh minh hoạ kiến trúc Tacotron 2 (2)

Kiến trúc này hoạt động như sau:

Ví dụ: Đoạn văn bản đầu vào là "My name is …" sẽ được chuyển thành các chuỗi âm vị như sau: /m/ /a/ ɪ/ /n/ /e/ /ɪ/ /m/ /ɪ/ /z/ … → **Character Embedding** (Các ký tự được nhúng dưới dạng 1 vector 512 chiều)

→ **3 convolution Layers 1D** (3 lớp tích chập 1 chiều) → sử dụng hàm kích hoạt **Relu** (làm cho dữ liệu sẽ được hội tụ lại từ đó làm tăng độ chính xác cho quá trình huấn luyện).

→ **Bi-directional LSTM** (512 neurons) → **encoded features** (Các đặc trưng được mã hoá) → **Concat** (Kết hợp với đặc trưng của giọng nói đích)

→ **Location Sensitive Attention** (Học tập các đặc trưng ở các lớp phía trước và cả của lớp hiện tại)

→ **2 LSTM layers** (1024 neurons) → **Linear Transform** (Biến đổi tuyến tính) → **Predicted Spectrogram Frame** (Dự đoán khung hình quang phổ)

→ **PostNet** (**5 Convolutional Layers**) → **Enhanced Prediction** (Dự đoán nâng cao)

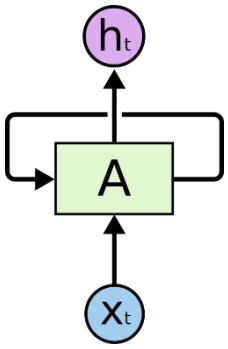
→ **modified Wavenet** (Mạng **Wavenet** đã được sửa đổi và được mang đi tổng hợp lại lần cuối).

Để có thể xử lý dữ liệu tuần tự một cách hiệu quả, chúng ta cần phải nói đến kiến trúc **LSTM** (**Long Short Term Memory**), nhưng do **LTSM** là một dạng kiến trúc đặc biệt của **RNN** (**Recurrent Neural Networks**: Mạng thần kinh tái phát) nên chúng sẽ điểm sơ qua kiến trúc này, sau đó sẽ đến kiến trúc **LSTM**.

Do con người không suy nghĩ lại từ đầu mỗi giây. Ví dụ: Khi chúng ta đọc một đoạn văn nào đó, chúng ta sẽ hiểu từng từ dựa trên sự hiểu biết của chúng ta đối với những từ trước đó. Sự hiểu biết của chúng ta có tính liên tục và bền bỉ.

Nhưng đối với mạng nơ ron truyền thống thì không thể làm được điều này, và điều này có lẽ là một thiếu sót lớn. Ví dụ: hãy tưởng tượng bạn muốn phân loại các sự kiện nào đang xảy ra tại mọi thời điểm trong một bộ phim. Không rõ bằng cách nào một mạng nơ-ron truyền thống có thể sử dụng lý luận của nó về các sự kiện trước đó trong phim để thông báo cho những sự kiện sau đó.

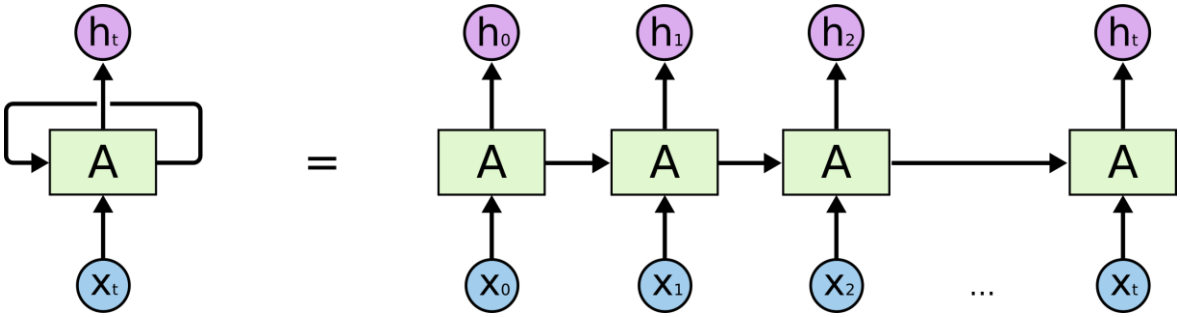
RNN có thể giải quyết vấn đề này. Chúng là một mạng với nhiều vòng lặp trong đó, cho phép thông tin có tính bền bỉ.



Hình 8. Minh hoạ 1 vòng lặp của kiến trúc RNN

Sơ đồ trên là một phần của mạng **A,** xét đầu vào là $\mathbf{x_t}$ và đầu ra là giá trị $\mathbf{h_t}$. Một vòng lặp cho phép thông tin được truyền từ một bước của mạng sang bước tiếp theo.

Những vòng lặp này làm cho các mạng lưới thần kinh tái phát có vẻ bí ẩn. Tuy nhiên, nếu bạn nghĩ nhiều hơn một chút, hóa ra chúng không khác gì một mạng lưới thần kinh bình thường. Một mạng thần kinh tái phát có thể được coi là nhiều bản sao của cùng một mạng, mỗi bản chuyển một thông điệp cho phần kế nhiệm. Hãy xem xét điều gì sẽ xảy ra nếu chúng ta hủy kiểm soát vòng lặp:



Hình 9. Mạng lưới RNN không được kiểm soát

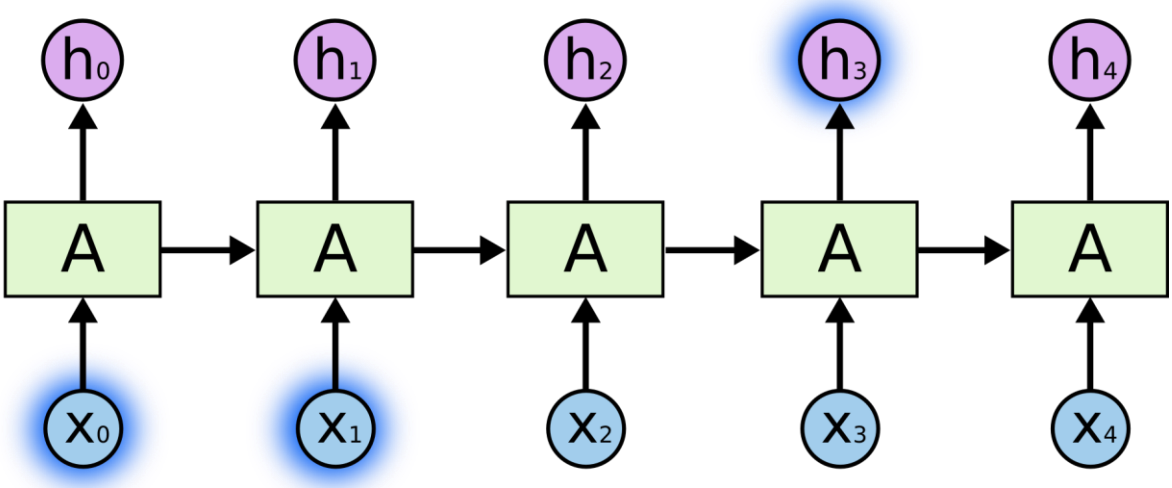
Bản chất giống như chuỗi này tiết lộ rằng các mạng thần kinh tái phát có liên quan mật thiết đến các chuỗi và danh sách. Chúng là kiến trúc tự nhiên của mạng lưới thần kinh để sử dụng cho những dữ liệu đó.

Và chúng chắc chắn được sử dụng! Trong vài năm gần đây, đã có những thành công đáng kinh ngạc khi áp dụng **RNN** cho nhiều vấn đề khác nhau: nhận dạng giọng nói, mô hình hóa ngôn ngữ, dịch thuật, chú thích hình ảnh… Danh sách này vẫn tiếp tục kéo dài.

Yếu tố cần thiết cho những thành công này là việc sử dụng "**LSTM**", một loại mạng thần kinh hồi quy rất đặc biệt, hoạt động đối với nhiều tác vụ tốt hơn nhiều so với phiên bản tiêu chuẩn. Hầu như tất cả các kết quả thú vị dựa trên mạng thần kinh tái phát đều đạt được với chúng. Đó là những **LSTM** mà chúng ta sẽ khám phá.

Một trong những điểm hấp dẫn của RNN là ý tưởng rằng chúng có thể kết nối thông tin trước đó với tác vụ hiện tại, chẳng hạn như việc sử dụng các khung hình video trước đó có thể giúp hiểu được khung hình hiện tại. Nếu **RNN** có thể làm được điều này, chúng sẽ cực kỳ hữu ích.

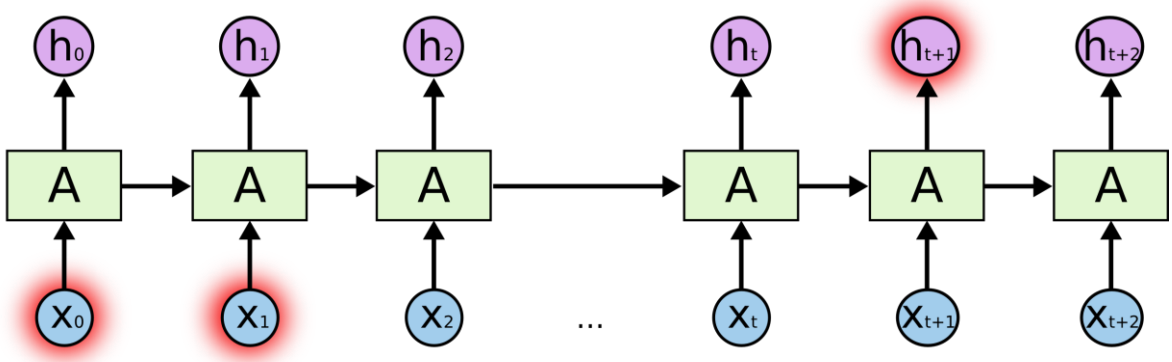
Đôi khi, chúng ta chỉ cần xem thông tin gần đây để thực hiện nhiệm vụ hiện tại. Ví dụ: hãy xem xét một mô hình ngôn ngữ đang cố gắng dự đoán từ tiếp theo dựa trên những từ trước đó. Nếu chúng ta đang cố gắng dự đoán từ cuối cùng trong "những đám mây đang ở trên bầu trời", thì chúng ta không cần thêm bất kỳ ngữ cảnh nào nữa khá rõ ràng là từ tiếp theo sẽ là "bầu trời". Trong những trường hợp như vậy, khi khoảng cách giữa thông tin liên quan và nơi cần đến là nhỏ, **RNN** có thể học cách sử dụng thông tin trong quá khứ.



Hình 10. Minh hoạ kiến trúc RNN (1)

Nhưng cũng có trường hợp chúng ta cần thêm ngữ cảnh. Cân nhắc việc cố gắng dự đoán từ cuối cùng trong văn bản "Tôi lớn lên ở Pháp… Tôi nói tiếng Pháp trôi chảy." Thông tin gần đây gợi ý rằng từ tiếp theo có thể là tên của một ngôn ngữ, nhưng nếu chúng ta muốn thu hẹp ngôn ngữ nào, chúng ta cần ngữ cảnh của Pháp, từ xa hơn trở lại. Khoảng cách giữa thông tin liên quan và điểm cần nó trở nên rất lớn là điều hoàn toàn có thể xảy ra.

Thật không may, khi khoảng cách đó tăng lên, **RNN** không thể học cách kết nối thông tin.



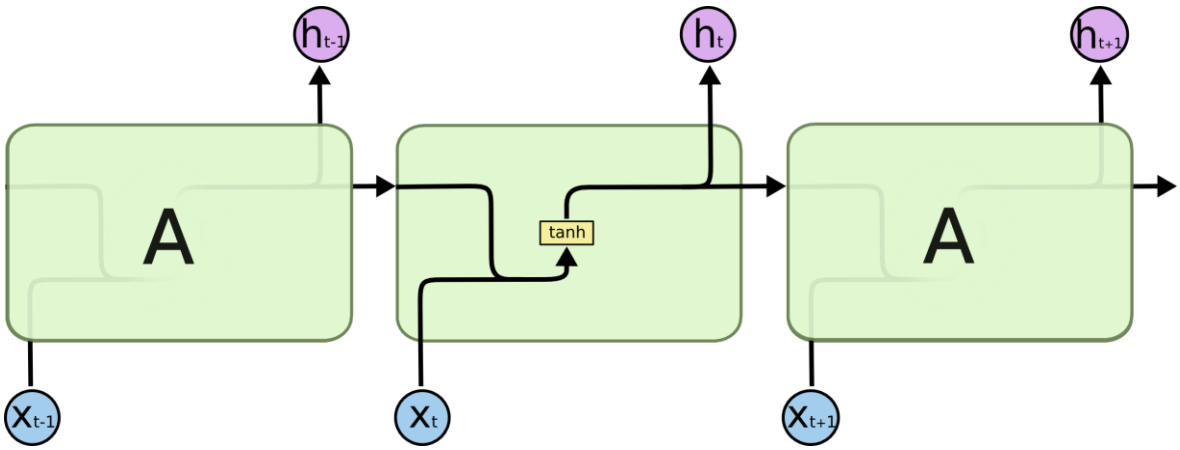
Hình 11. Minh hoạ kiến trúc RNN (2)

Về lý thuyết, **RNN** hoàn toàn có khả năng xử lý "sự phụ thuộc dài hạn" như vậy. Con người có thể cẩn thận chọn các thông số để chúng giải quyết các bài toán dạng này. Đáng buồn thay, trên thực tế, **RNN** dường như không thể học chúng. Vấn đề đã được khám phá sâu bởi Hochreiter (1991) [tiếng Đức] và Bengio, et al. (1994), người đã tìm ra một số lý do khá cơ bản giải thích tại sao nó có thể khó khăn. Rất may, **LSTM** không gặp vấn đề này!

Mạng **Long Short Term Memory** thường được gọi là "**LSTM**" - là một loại **RNN** đặc biệt, có khả năng học các phụ thuộc dài hạn. Chúng được giới thiệu bởi Hochreiter & Schmidhuber (1997), và được nhiều người chắt lọc, phổ biến trong các tác phẩm sau. Chúng hoạt động rất hiệu quả trong nhiều vấn đề khác nhau và hiện đang được sử dụng rộng rãi.

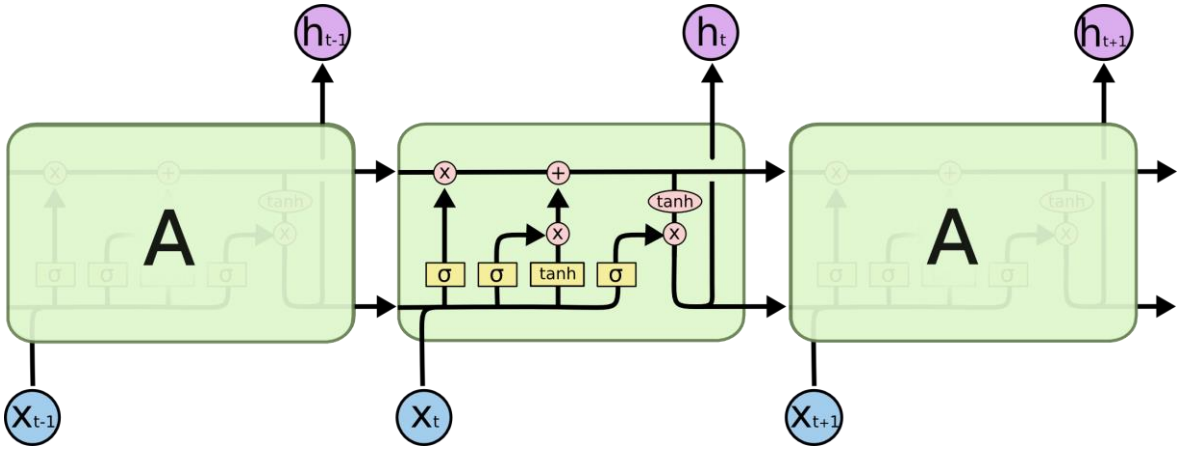
LSTM được thiết kế rõ ràng để tránh vấn đề phụ thuộc lâu dài. Ghi nhớ thông tin trong thời gian dài thực tế là hành vi mặc định của họ, không phải là thứ mà họ phải vật lộn để học!

Tất cả các mạng thần kinh tái phát đều có dạng một chuỗi các mô-đun lặp lại của mạng thần kinh. Trong các **RNN** tiêu chuẩn, mô-đun lặp lại này sẽ có cấu trúc rất đơn giản, chẳng hạn như một lớp tanh.



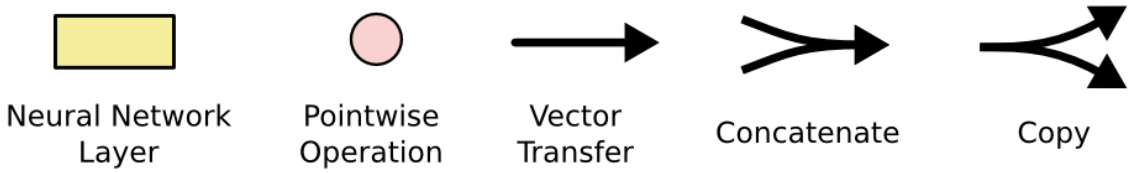
Hình 12. Minh hoạ kiến trúc LSTM (1)

Các **LSTM** cũng có cấu trúc giống như chuỗi này, nhưng mô-đun lặp lại có cấu trúc khác. Thay vì có một lớp mạng thần kinh duy nhất, có bốn lớp, tương tác theo một cách rất đặc biệt.



Hình 13. Minh hoạ kiến trúc LSTM (2)

Đừng lo lắng về các chi tiết của những gì đang xảy ra. Chúng ta sẽ xem sơ đồ LSTM từng bước sau. Hiện tại, chúng ta hãy cố gắng làm quen với ký hiệu mà chúng ta sẽ sử dụng.

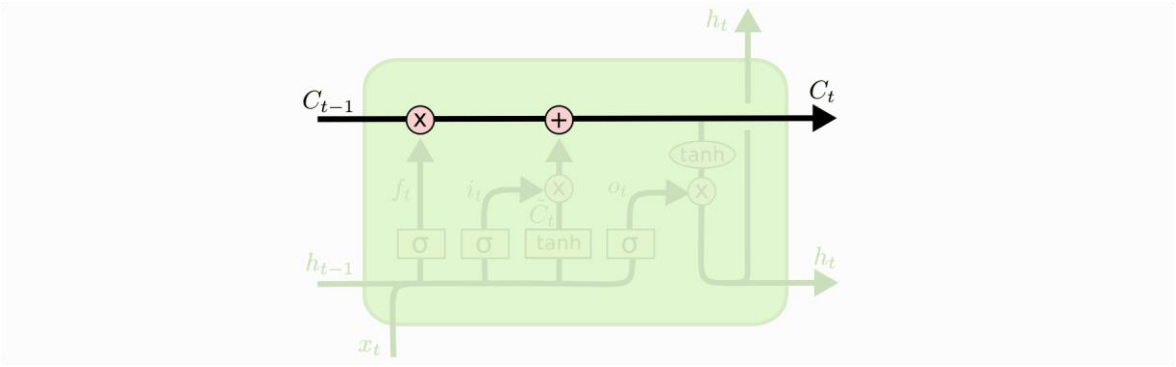


Hình 14. Các ký hiệu chung của mô hình LSTM

Trong sơ đồ trên, mỗi dòng mang toàn bộ một vectơ, từ đầu ra của một nút đến đầu vào của các nút khác. Các vòng tròn màu hồng biểu thị các hoạt động theo chiều kim đồng hồ, như phép cộng vectơ, trong khi các hộp màu vàng là các lớp mạng thần kinh đã học. Các dòng hợp nhất biểu thị sự nối, các dòng rẽ nhánh biểu thị nội dung của nó đang được sao chép và các bản sao sẽ đến các vị trí khác nhau.

Chìa khóa của **LSTM** là trạng thái ô, đường nằm ngang chạy qua đỉnh của sơ đồ.

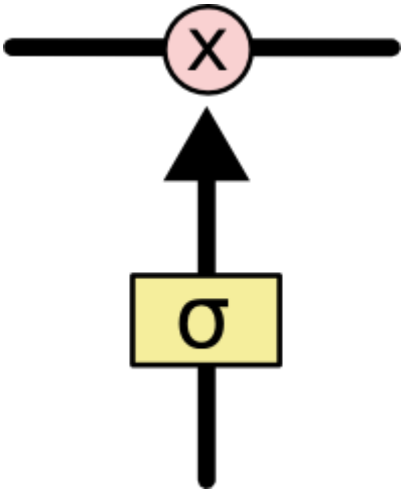
Trạng thái tế bào giống như một băng chuyền. Nó chạy thẳng xuống toàn bộ chuỗi, chỉ với một số tương tác tuyến tính nhỏ. Rất dễ để thông tin cứ trôi theo nó mà không thay đổi.



Hình 15. Minh hoạ kiến trúc LSTM (3)

LSTM có khả năng loại bỏ hoặc thêm thông tin vào trạng thái tế bào, được điều chỉnh cẩn thận bởi các cấu trúc gọi là cổng.

Cổng là một cách để tùy ý cho thông tin đi qua. Chúng bao gồm một lớp lưới thần kinh sigmoid và phép toán nhân theo điểm.



Hình 16. Cổng kiểm tra thông tin kiến trúc LSTM

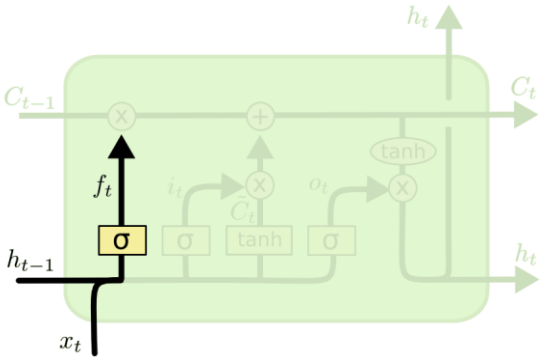
Lớp sigmoid xuất ra các số từ 0 đến 1, mô tả mức độ cho phép của mỗi thành phần. Giá trị bằng 0 có nghĩa là "không để gì lọt qua", trong khi giá trị bằng 1 có nghĩa là "để mọi thứ lọt qua!"

Một **LSTM** có ba trong số các cổng này, để bảo vệ và kiểm soát trạng thái tế bào.

Bước đầu tiên trong **LSTM** của chúng ta là quyết định thông tin nào chúng ta sẽ loại bỏ khỏi trạng thái tế bào. Quyết định này được thực hiện bởi một lớp sigmoid được gọi là "lớp cổng quên". Nó xem $\mathbf{h_{t-1}}$ và $\mathbf{x_t}$, và xuất ra một số từ **0** đến **1** cho mỗi

số ở trạng thái ô $\mathbf{C_{t-1}}$. Số **1** đại diện cho "hoàn toàn giữ điều này" trong khi số **0** đại diện cho "hoàn toàn loại bỏ điều này".

Hãy quay lại ví dụ của chúng ta về một mô hình ngôn ngữ đang cố gắng dự đoán từ tiếp theo dựa trên tất cả các từ trước đó. Trong một vấn đề như vậy, trạng thái ô có thể bao gồm giới tính của chủ thể hiện tại, để có thể sử dụng đại từ chính xác. Khi nhìn thấy một chủ đề mới, chúng ta muốn quên đi giới tính của chủ đề cũ.

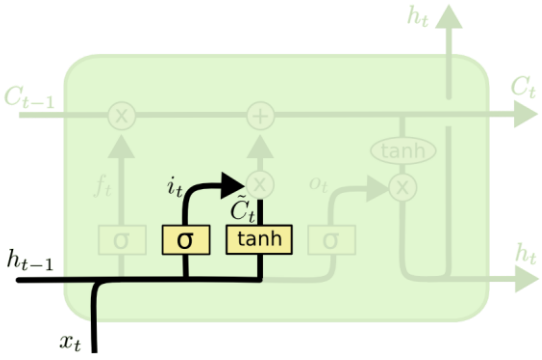


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Hình 17. Minh hoạ kiến trúc LSTM (4)

Bước tiếp theo là quyết định thông tin mới nào chúng ta sẽ lưu trữ ở trạng thái ô. Điều này có hai phần. Đầu tiên, một lớp sigmoid được gọi là "lớp cổng đầu vào" quyết định giá trị nào chúng ta sẽ cập nhật. Tiếp theo, một lớp tanh tạo ra một vectơ các giá trị ứng viên mới, $\widetilde{\boldsymbol{C}}_{\mathbf{t}}$, có thể được thêm vào trạng thái. Trong bước tiếp theo, chúng ta sẽ kết hợp cả hai điều này để tạo một bản cập nhật cho trạng thái.

Trong ví dụ về mô hình ngôn ngữ của chúng ta, chúng ta muốn thêm giới tính của chủ thể mới vào trạng thái ô, để thay thế cái cũ mà chúng ta đang quên.



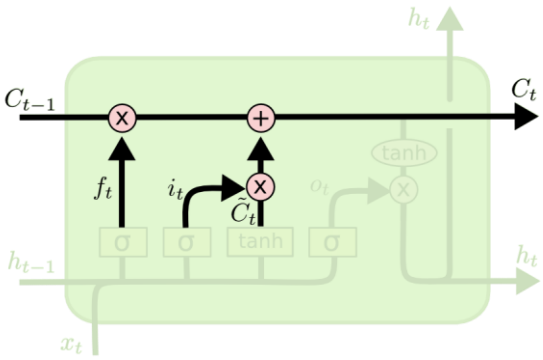
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Hình 18. Minh hoạ kiến trúc LSTM (5)

Đã đến lúc cập nhật trạng thái ô cũ, $\mathbf{C_{t-1}}$, sang trạng thái ô mới $\mathbf{C_t}$. Các bước trước đã quyết định phải làm gì, chúng ta chỉ cần thực sự làm điều đó.

Ta nhân đôi tình cũ $\mathbf{f_t}$, quên đi những điều ta đã định quên trước đó. Sau đó, chúng ta thêm $\mathbf{i_t * \tilde{C}_t}$. Đây là giá trị ứng cử viên mới, được chia tỷ lệ theo mức độ chúng ta quyết định cập nhật từng giá trị trạng thái.

Trong trường hợp của mô hình ngôn ngữ, đây là nơi chúng ta thực sự loại bỏ thông tin về giới tính của đối tượng cũ và thêm thông tin mới, như chúng ta đã quyết định trong các bước trước.

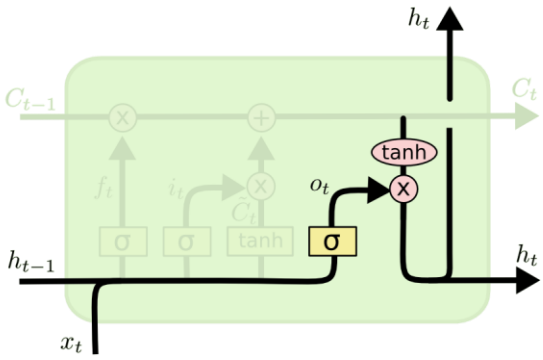


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Hình 19. Minh hoạ kiến trúc LSTM (6)

Cuối cùng, chúng ta cần quyết định những gì chúng ta sẽ xuất ra. Đầu ra này sẽ dựa trên trạng thái ô của chúng ta, nhưng sẽ là phiên bản được lọc. Đầu tiên, chúng ta chạy một lớp sigmoid quyết định phần nào của trạng thái ô mà chúng ta sẽ xuất ra. Sau đó, chúng ta đặt trạng thái ô thông qua **tanh** (để đẩy các giá trị nằm trong khoảng từ **-1** đến **1**) và nhân nó với đầu ra của cổng sigmoid, để chúng ta chỉ xuất ra những phần mà chúng ta đã quyết định.

Đối với ví dụ về mô hình ngôn ngữ, vì nó chỉ nhìn thấy một chủ đề nên nó có thể muốn xuất thông tin liên quan đến một động từ, trong trường hợp đó là điều tiếp theo. Ví dụ: nó có thể xuất ra chủ ngữ là số ít hay số nhiều, để chúng ta biết động từ nên được chia thành dạng nào nếu đó là điều tiếp theo.

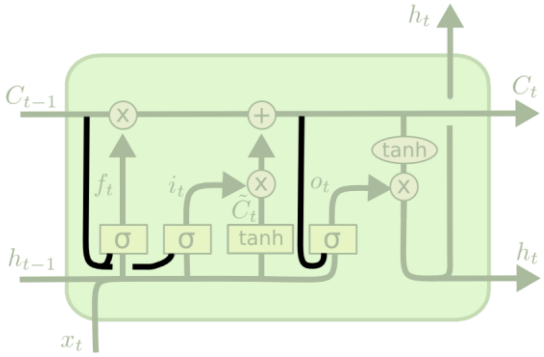


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Hình 20. Minh hoạ kiến trúc LSTM (7)

Những gì ta đã mô tả cho đến nay là một **LSTM** khá bình thường. Nhưng không phải tất cả các **LSTM** đều giống như trên. Trên thực tế, có vẻ như hầu hết mọi bài báo liên quan đến **LSTM** đều sử dụng một phiên bản hơi khác. Sự khác biệt là nhỏ, nhưng đáng nói đến một số trong số họ.

Một biến thể **LSTM** phổ biến, được giới thiệu bởi Gers & Schmidhuber (2000), đang thêm "kết nối lỗ nhìn trộm". Điều này có nghĩa là chúng ta để các lớp cổng xem xét trạng thái của ô.

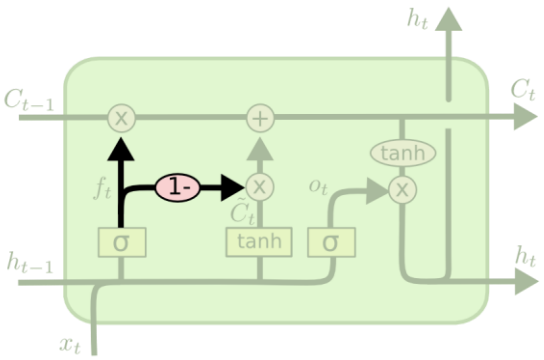


$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

Hình 21. Minh hoạ kiến trúc LSTM (8)

Sơ đồ trên thêm các lỗ nhìn trộm vào tất cả các cổng, nhưng nhiều bài báo sẽ cung cấp một số lỗ nhìn trộm chứ không phải các lỗ khác.

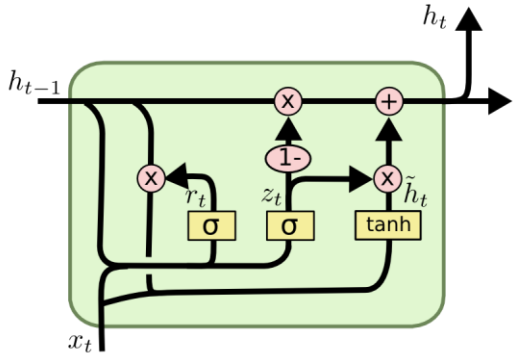
Một biến thể khác là sử dụng các cổng nhập và quên kết hợp. Thay vì quyết định riêng những gì cần quên và những gì chúng ta nên thêm thông tin mới vào, chúng ta đưa ra những quyết định đó cùng nhau. Chúng ta chỉ quên khi chúng ta sẽ nhập một cái gì đó vào vị trí của nó. Chúng ta chỉ nhập các giá trị mới vào trạng thái khi chúng ta quên thứ gì đó cũ hơn.



$$C_t = f_t * C_{t-1} + (1 - \boldsymbol{f_t}) * \tilde{C}_t$$

Hình 22. Minh hoạ kiến trúc LSTM (9)

Một biến thể ấn tượng hơn một chút trên LSTM là Gated Recurrent Unit, hay GRU, được giới thiệu bởi Cho, et al. (2014). Nó kết hợp các cổng nhập và quên thành một "cổng cập nhật" duy nhất. Nó cũng hợp nhất trạng thái ô và trạng thái ẩn, đồng thời thực hiện một số thay đổi khác. Mô hình kết quả đơn giản hơn các mô hình LSTM tiêu chuẩn và ngày càng trở nên phổ biến.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Hình 23. Minh hoạ kiến trúc LSTM (10)

Đây chỉ là một số biến thể LSTM đáng chú ý nhất. Có rất nhiều thứ khác, như Depth Gated RNNs của Yao, et al. (2015). Ngoài ra còn có một số cách tiếp cận hoàn toàn khác để giải quyết các phụ thuộc dài hạn, như Clockwork RNNs của Koutnik, et al. (2014).

Biến thể nào trong số này là tốt nhất? Sự khác biệt có quan trọng không? Greff, et al. (2015) đã thực hiện một so sánh thú vị về các biến thể phổ biến, nhận thấy rằng tất cả chúng đều giống nhau. Jozefowicz, et al. (2015) đã thử nghiệm hơn mười nghìn kiến trúc RNN, tìm thấy một số kiến trúc hoạt động tốt hơn LSTM trong một số tác vụ nhất định.

Hiện tại, việc triển khai chính thức các dự án của SV2TTS vẫn chưa được công khai. Tuy nhiên, vẫn có những triển khai khác do cộng đồng xây dựng nên. CorentinJ/Real-Time-Voice-Cloning là dự án tốt nhất hiện có được công khai. Dự án được phát triển bởi Corentin Jemine, người đã có bằng Thạc sĩ Khoa học Dữ liệu tại Đại học Liège. Nó đi kèm với một giao diện người dùng có thể được sử dụng để tạo âm thanh.

3. Các thư viện và công cụ hỗ trợ

3.1. Librosa

Librosa là một gói Python để phân tích âm nhạc và âm thanh. Nó cung cấp các khối xây dựng cần thiết để tạo ra các hệ thống truy xuất thông tin âm nhạc.

3.2. PyQt5

Qt là một Application framework đa nền tảng viết trên ngôn ngữ C++ , được dùng để phát triển các ứng dụng trên desktop, hệ thống nhúng và mobile. Hỗ trợ cho các platform bao gồm : Linux, OS X, Windows, VxWorks, QNX, Android, iOS, BlackBerry, Sailfish OS và một số platform khác. PyQt là Python interface của Qt, kết hợp của ngôn ngữ lập trình Python và thư viện Qt, là một thư viện bao gồm các thành phần giao diện điều khiển (**widgets** , **graphical control elements**).

3.3. Pillow

Pillow là một fork từ thư viện PIL của Python được sử dụng để xử lý hình ảnh. Pillow có thể đọc, ghi hình ảnh, cắt, dán, nhập ảnh, biến đổi về mặt hình học hoặc kể cả biến đổi màu sắc của ảnh.

3.4. Inflect

Tạo chính xác danh từ số nhiều, số ít, thứ tự, mạo từ không xác định hoặc có thể chuyển số thành chữ. Ví dụ: Số ít của “them” là “it”.

3.5. Pytorch

PyTorch là một framework được xây dựng dựa trên python cung cấp nền tảng tính toán khoa học phục vụ lĩnh vực Deep learning. Pytorch tập trung vào 2 khả năng chính:

- Một sự thay thế cho bộ thư viện numpy để tận dụng sức mạnh tính toán của GPU.

- Một platform Deep learning phục vụ trong nghiên cứu, mang lại sự linh hoạt và tốc độ.

❖ Ưu điểm:

- Mang lại khả năng debug dễ dàng hơn theo hướng interactively, rất nhiều nhà nghiên cứu và kỹ sư đã dùng cả pytorch và tensorflow đều đánh giá cao pytorch hơn trong vấn đề debug và visualize.
- Hỗ trợ tốt dynamic graphs.
- Được phát triển bởi đội ngũ Facebook.
- Kết hợp cả các API cấp cao và cấp thấp.

❖ Nhược điểm:

- Vẫn chưa được hoàn thiện trong việc deploy, áp dụng cho các hệ thống lớn,... được như framework ra đời trước nó như tensorflow.
- Ngoài document chính từ pytorch thì vẫn còn khá hạn chế các nguồn tài liệu bên ngoài như các tutorials hay các câu hỏi trên stackoverflow.

3.6. Ffmpeg

FFmpeg là một framework hàng đầu về đa phương tiện (xử lý audio, video). Nó có thể decode (giải mã), encode (mã hóa), transcode (chuyển mã), mux (ghép kênh), demux (phân kênh, tách kênh), stream (ví dụ như livestream trên youtube, facebook,..), filter (lọc) và play (chạy, phát video) rất nhiều thứ mà con người hay máy móc tạo ra.

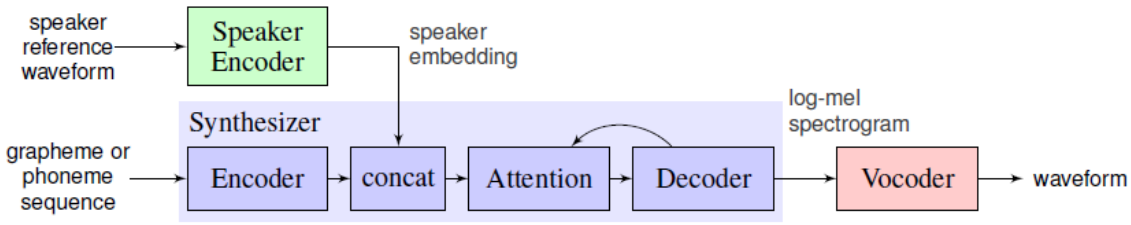
FFmpeg hỗ trợ hầu hết các định dạng. Và nó khá là linh hoạt, có thể compile, run và chạy trên nhiều nền tảng như Linux, Mac OS X, Microsoft Windows, BSD, Solaris,...và ở trên nhiều môi trường, kiến trúc khác nhau.

3.7. Webrtcvad

Đây là giao diện python cho Trình phát hiện hoạt động bằng giọng nói WebRTC (VAD). VAD dùng để phân loại một đoạn dữ liệu âm thanh là có tiếng hoặc không có tiếng. Nó có thể hữu ích cho điện thoại và nhận dạng giọng nói. VAD mà Google phát triển cho dự án WebRTC được cho là một trong những VAD tốt nhất hiện có, nhanh, hiện đại và miễn phí.

CHƯƠNG 2
THIẾT KẾ VÀ CÀI ĐẶT

1. Thiết kế hệ thống nhân bản giọng nói

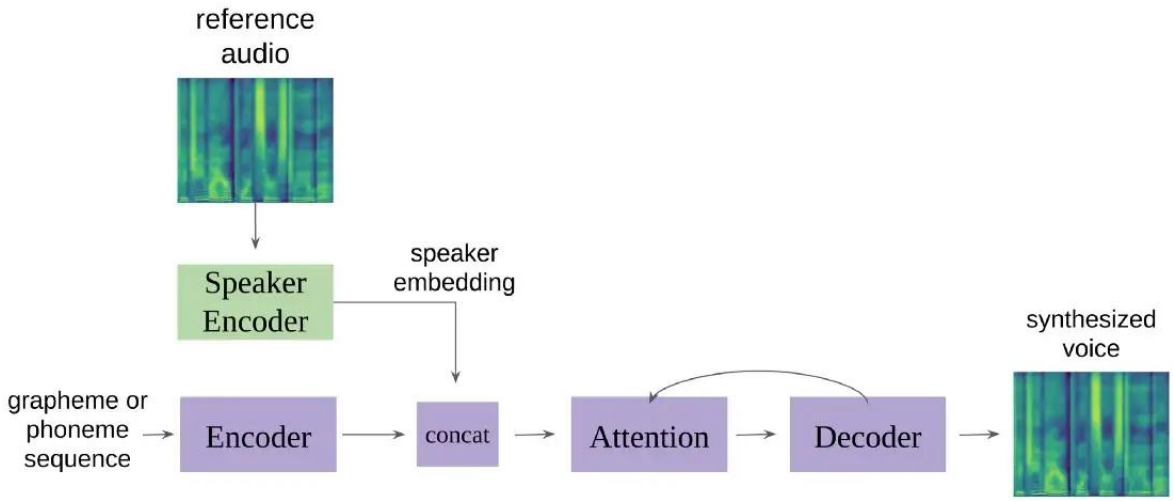


Hình 24. Mô hình hệ thống SV2TTS

Hình 24 mô tả các thành phần cốt lõi của ứng dụng nhân bản giọng nói bao gồm:
Bộ mã hoá giọng nói (Speaker Encoder): Công việc của bộ mã hóa giọng nói là lấy một số âm thanh đầu vào (được mã hóa dưới dạng quang phổ mel) của một giọng nói nhất định và xuất ra một bản nhúng ghi lại "âm thanh của người đó phát ra như thế nào". Bộ mã hoá âm thanh không quan tâm đến những từ ngữ mà người đó đang nói hoặc bất kì âm thanh ồn nào trong đoạn audio, những gì nó quan tâm đến là giọng nói của người đang nói. Ví dụ: giọng nói cao/thấp, trọng âm, giai điệu,… Tất cả các thuộc tính này được kết hợp thành một vector chiều thấp, được gọi chính thức là d-vector hoặc bộ nhúng âm thanh.
Quy trình để tạo ra các phần nhúng trên được mô tả như sau:

- Đầu tiên, các ví dụ về âm thanh lời nói được phân đoạn thành các clip dài 1,6 giây không có bản phiên âm và được chuyển đổi thành quang phổ mel.
- Sau đó, bộ mã hóa giọng nói của người nói được huấn luyện để lấy hai mẫu âm thanh và quyết định xem có phải do cùng một người nói tạo ra chúng hay không. Là một sản phẩm phụ, điều này buộc bộ mã hóa người nói tạo ra các phần nhúng thể hiện âm thanh của người nói.

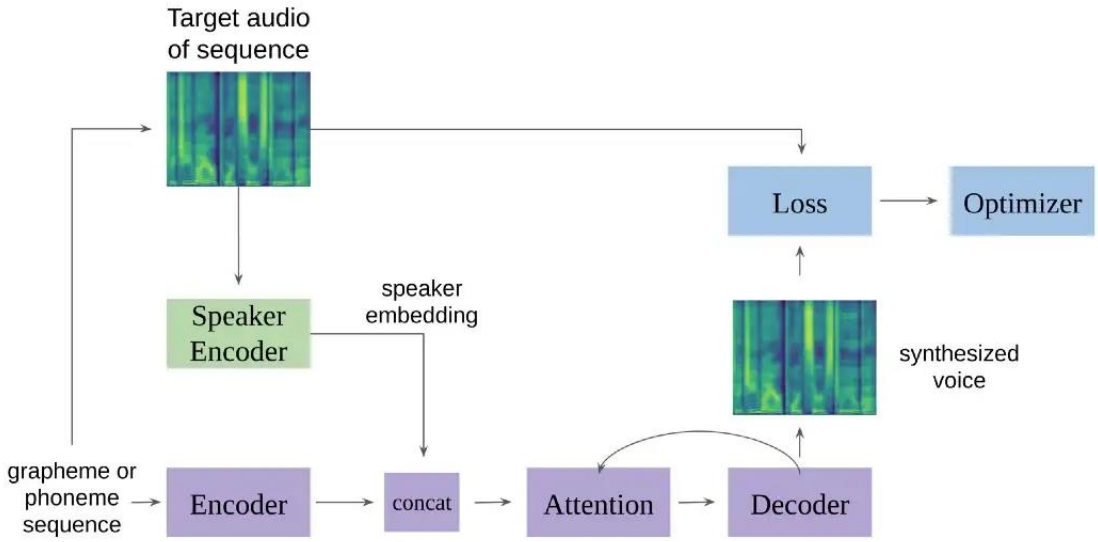
Bộ tổng hợp (Synthesizer): Bộ tổng hợp là một phần của SV2TTS phân tích đầu vào văn bản để tạo quang phổ mel, sau đó bộ mã hóa giọng nói sẽ chuyển đổi thành âm thanh. Bộ tổng hợp có một chuỗi văn bản - được ánh xạ tới các âm vị (đơn vị nhỏ nhất của âm thanh người, ví dụ: âm thanh bạn tạo ra khi nói 'a'), cùng với các phần nhúng do bộ mã hóa người nói tạo ra và sử dụng kiến trúc **Tacotron 2** để tạo các khung của biểu đồ quang phổ mel một cách lặp lại.



Hình 25. Kiến trúc Tacotron 2.

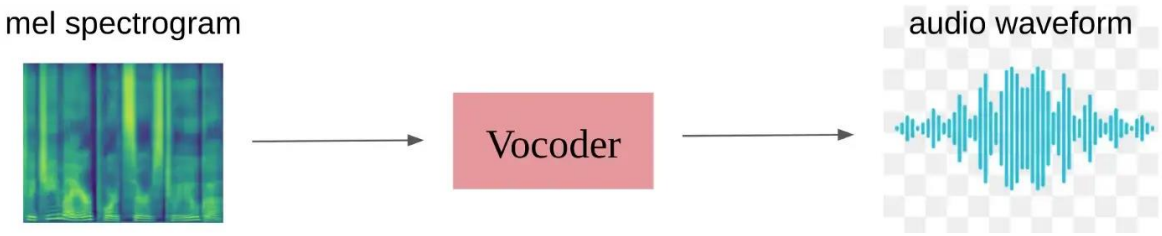
Để đào tạo bộ tổng hợp, chúng ta cần:

- Đầu tiên, chúng ta thu thập một chuỗi các âm vị (phoneme sequence) và quang phổ mel của người nói phát ra câu đó.
- Sau đó, quang phổ mel được chuyển đến bộ mã hóa người nói để tạo các giọng nói được nhúng.
- Tiếp theo, bộ mã hóa (encoder) của bộ tổng hợp nối (concat) mã hóa chuỗi âm vị (phoneme sequence) của nó với phần nhúng của người nói.
- Biểu đồ quang phổ mel được tạo định kỳ bởi bộ giải mã (decoder) và bộ phận chú ý (attention) của bộ tổng hợp (synthesizer)
- Cuối cùng, phổ mel được so sánh với mục tiêu ban đầu để tạo ra sự mất mát, sau đó được tối ưu hóa.



Hình 26. Đào tạo mạng tổng hợp (Synthesizer Training)

Bộ phát âm (Vocoder): Tại thời điểm này, bộ tổng hợp đã tạo ra một phổ mel, nhưng chúng ta vẫn chưa thể nghe bất cứ điều gì. Để chuyển đổi phổ mel thành sóng âm thanh thô, các tác giả sử dụng bộ phát âm. Bộ phát âm đặc biệt được sử dụng ở đây dựa trên mô hình WaveNet của DeepMind, mô hình này tạo ra các dạng sóng âm thanh thô từ văn bản và có thời điểm là công nghệ tiên tiến nhất dành cho các hệ thống TTS.

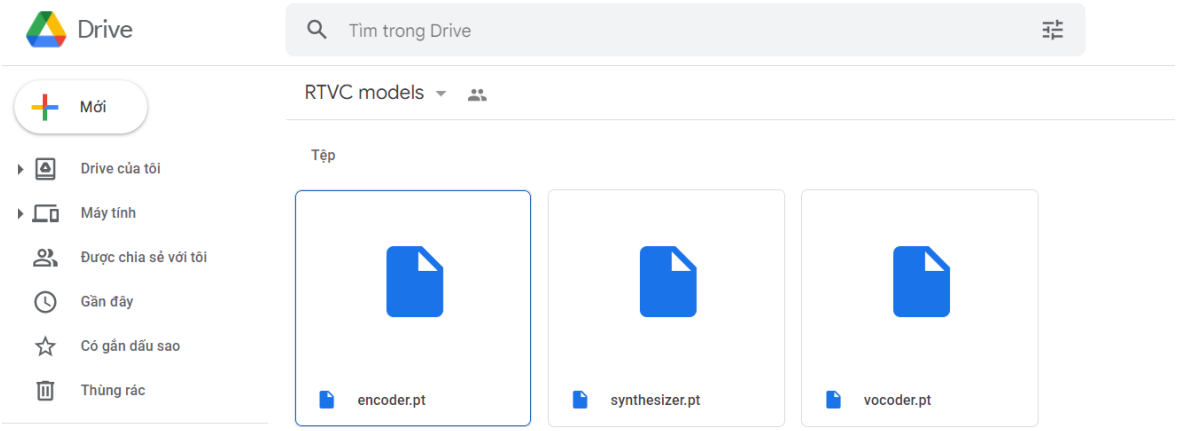


Hình 27. Quá trình chuyển đổi từ quang phổ mel sang audio

2. Xây dựng hệ thống

2.1. Cài đặt mô hình đã được đào tạo sẵn

Chúng ta truy cập đường link [1] để dẫn đến thư mục có chứa các file được đào tạo trước:



Hình 28. Ảnh minh hoạ mô hình được tào tạo sẵn

Mô hình đã sử dụng hai bộ dữ liệu công khai để đào tạo mạng tổng hợp giọng nói và bộ mã hóa giọng nói. **VCTK (Voice Cloning Tool Kit: bộ công cụ nhân bản giọng nói)** chứa 44 giờ bài phát biểu rõ ràng từ 109 người nói, phần lớn trong số đó có giọng Anh. Mô hình này đã giảm lấy mẫu âm thanh xuống 24 kHz, cắt bớt đoạn không có âm thanh ở đầu và cuối (giảm thời lượng trung bình từ 3,3 giây xuống 1,8 giây) và chia thành ba tập hợp con: đào tạo, xác thực (chứa các giọng nói của diễn giả giống như bộ đào tạo) và kiểm tra (chứa 11 diễn giả được đưa ra từ bộ đào tạo và bộ xác thực).

LibriSpeech bao gồm sự kết hợp của hai bộ đào tạo "sạch", bao gồm 436 giờ phát biểu từ 1.172 người nói, được lấy mẫu ở tần số 16 kHz. Phần lớn bài phát biểu là tiếng Anh Mỹ, tuy nhiên vì nó có nguồn gốc từ sách nói, giọng điệu và phong cách nói có thể khác biệt đáng kể giữa các phát biểu của cùng một người nói. Mô hình đã phân đoạn lại dữ liệu thành những lời ngắn hơn bằng cách căn chỉnh âm thanh với bản ghi bằng cách sử dụng mô hình ASR và ngắt các phân đoạn khi im lặng, giảm thời lượng trung bình từ 14 giây xuống 5 giây. Như trong tập dữ liệu gốc, không có dấu chấm câu trong bảng điểm. Các bộ âm thanh người nói hoàn toàn rời rạc giữa các bộ đào tạo, xác nhận và thử nghiệm.

Nhiều bản ghi trong kho dữ liệu sạch của **LibriSpeech** chứa tiếng ồn xung quanh môi trường và tĩnh đáng chú ý. Mô hình đã xử lý trước chương trình phổ mục tiêu bằng cách sử dụng quy trình trừ phổ đơn giản, trong đó phổ nhiễu nền của một câu nói được ước tính là phân vị thứ 10 của năng lượng trong mỗi dải tần trên toàn

bộ tín hiệu. Quá trình này chỉ được sử dụng trên mục tiêu tổng hợp; bài phát biểu chứa tiếng ồn ban đầu đã được chuyển đến bộ mã âm thanh người nói.

Mô hình đã được đào tạo mạng lưới tổng hợp và bộ mã hóa riêng biệt cho mỗi một trong hai kho tài liệu này. Trong suốt phần này, mô hình này đã sử dụng các mạng tổng hợp được đào tạo về đầu vào âm vị, để kiểm soát việc phát âm trong các đánh giá chủ quan. Đối với tập dữ liệu **VCTK**, có âm thanh khá rõ ràng, do đó tác giả đã nhận thấy rằng bộ phát âm được đào tạo về phổ mel sự thật hoạt động tốt. Tuy nhiên, đối với **LibriSpeech** thì chứa nhiều tiếng ồn hơn, tác giả nhận thấy cần phải đào tạo bộ mã hóa về các biểu đồ quang phổ được dự đoán bởi mạng tổng hợp. Không có nhiễu được thực hiện trên dạng sóng đích để đào tạo bộ mã hóa.

Bộ mã hóa âm thanh người nói đã được đào tạo trên kho ngữ liệu tìm kiếm bằng giọng nói độc quyền chứa 36 triệu câu nói với thời lượng trung bình 3,9 giây từ 18 nghìn người nói tiếng Anh ở Hoa Kỳ. Tập dữ liệu này không được phiên âm nhưng chứa danh tính người nói ẩn danh. Nó không bao giờ được sử dụng để đào tạo mạng tổng hợp.

Mô hình chủ yếu được dựa vào đánh giá Điểm ý kiến trung bình (MOS) do cộng đồng cung cấp dựa trên các bài kiểm tra nghe chủ quan. Tất cả các đánh giá MOS của chúng tôi đều phù hợp với thang điểm Xếp hạng loại tuyệt đối, với điểm xếp hạng từ 1 đến 5 với gia số 0,5 điểm. Chúng tôi sử dụng khung này để đánh giá bài phát biểu tổng hợp theo hai chiều: tính tự nhiên và độ giống với bài phát biểu thực của người nói đích.

2.2. Bộ mã hoá âm thanh người nói (Speaker Encoder)

- Cài đặt thư viện **webrtcvad** để loại bỏ đi các đoạn âm thanh không có tiếng nói**: pip install webrtcvad**

- Cài đặt thư viện **pytorch** để sử dụng các lớp cơ hỗ trợ về mạng nơ-ron: **pip install torch==1.12.0 torchvision==0.13.0 torchaudio==0.12.0**

Sau khi cài đặt thành công thư viện **pytorch** chúng ta sẽ xây dựng mô hình mạng nơ ron dựa trên ví dụ mà nhà phát triển đã hướng dẫn trên tài liệu trên trang chủ: [2]

Docs > torch.nn > Module

MODULE

CLASS torch.nn.Module [SOURCE]

Base class for all neural network modules.

Your models should also subclass this class.

Modules can also contain other Modules, allowing to nest them in a tree structure. You can assign the submodules as regular attributes:

```
import torch.nn as nn
import torch.nn.functional as F

class Model(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(1, 20, 5)
        self.conv2 = nn.Conv2d(20, 20, 5)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        return F.relu(self.conv2(x))
```

Hình 29. Cách sử dụng lớp cơ sở về mạng nơ ron của thư viện Pytorch

- Cài đặt thư viện **librosa** chuyển đổi dữ âm thanh sang quang phổ mel: **pip install librosa**

```
def wav_to_mel_spectrogram(wav):
    """
    Derives a mel spectrogram ready to be used by the encoder from a preprocessed audio waveform.
    Note: this not a log-mel spectrogram.
    """
    frames = librosa.feature.melspectrogram(
        wav,
        sampling_rate,
        n_fft=int(sampling_rate * mel_window_length / 1000),
        hop_length=int(sampling_rate * mel_window_step / 1000),
        n_mels=mel_n_channels
    )
    return frames.astype(np.float32).T
```

2.3. Bộ tổng hợp âm thanh từ người nói (Synthesizer)

Cài đặt kiến trúc **tacotron 2** được xây dựng sẵn từ pytorch bằng đường link sau: [3]

Sau khi đã cài đặt thành công kiến trúc trên ta tiến hành xây dựng các chức năng tổng hợp quang phổ mel từ đoạn văn bản và âm thành người nói đã được nhúng từ bộ mã hoá âm thanh người nói.

2.4. Bộ phát âm (Vocoder)

Cài đặt bộ phát âm để chuyển đổ quang phổ mel sang âm thanh sau khi được tổng hợp từ bộ tổng hợp phía trên dựa trên kiến trúc **fatchord** được xây dựng sẵn bằng đường link sau: [4]

Sau khi cài đặt kiến trúc thành công ta sẽ sử dụng nó để tạo ra âm thanh kết quả thu được và hoàn tất mô hình.

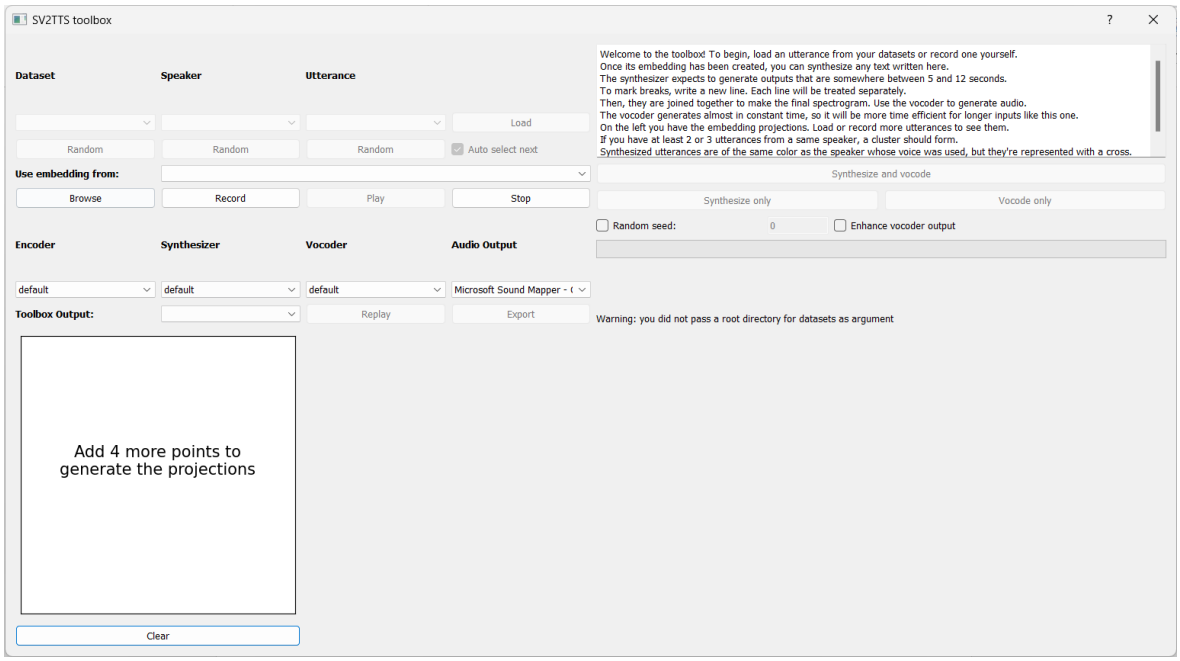
2.5. Giao diện (UI)

Cài đặt thư viện PyQt5: **pip install PyQt5**. Sau khi cài đặt thư viện thành công chúng ta sẽ xây dựng giao diện bằng các công cụ mà thư viện này đã hỗ trợ để có được giao diện như mong muốn. Tài liệu hỗ trợ xây dựng giao diện python bằng thư viện PyQt5 [5]

CHƯƠNG 3
KẾT QUẢ THỰC NGHIỆM

1. Giao diện sản phẩm

Sau khi cài đặt xong sản phẩm có giao diện như sau:

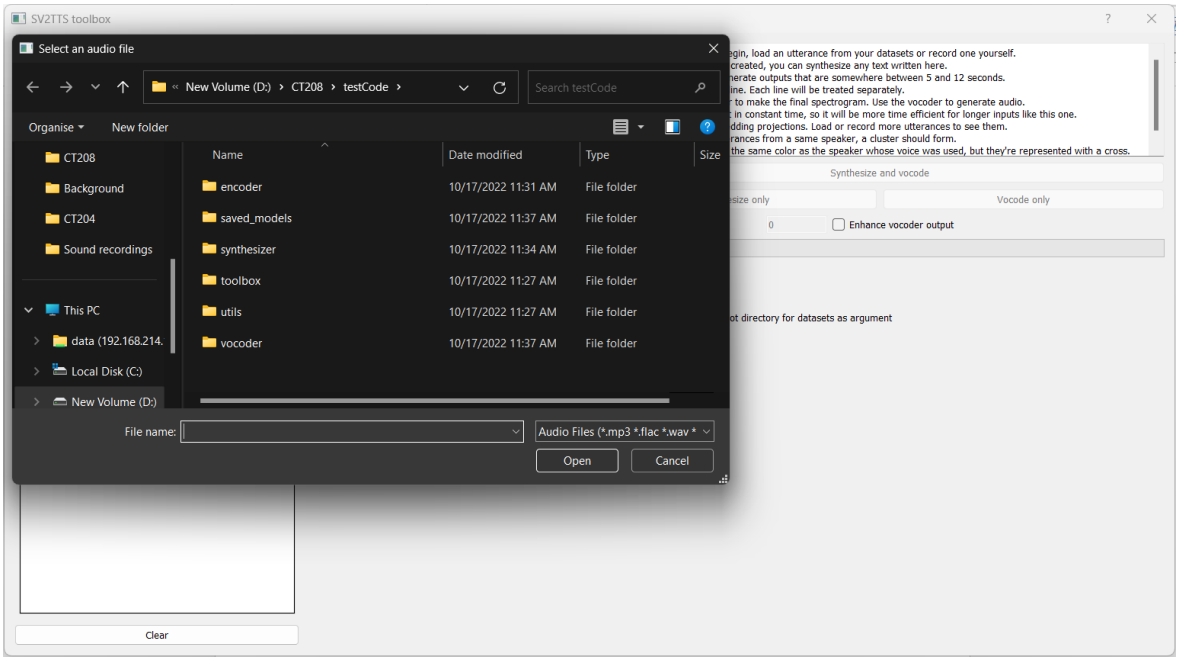


Hình 30. Giao diện kết quả (1)

2. Giới thiệu các tính năng

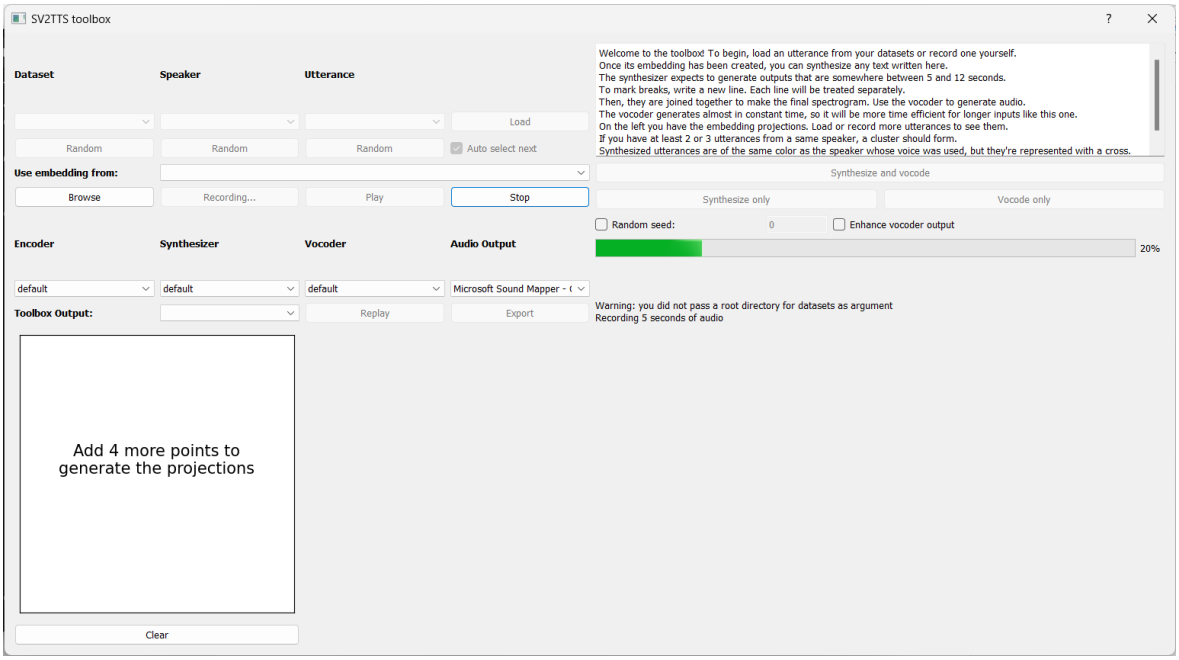
Sản phẩm bao gồm các chức năng như sau:

- Browse: Dùng để tải tệp âm thanh từ máy



Hình 31. Giao diện kết quả (2)

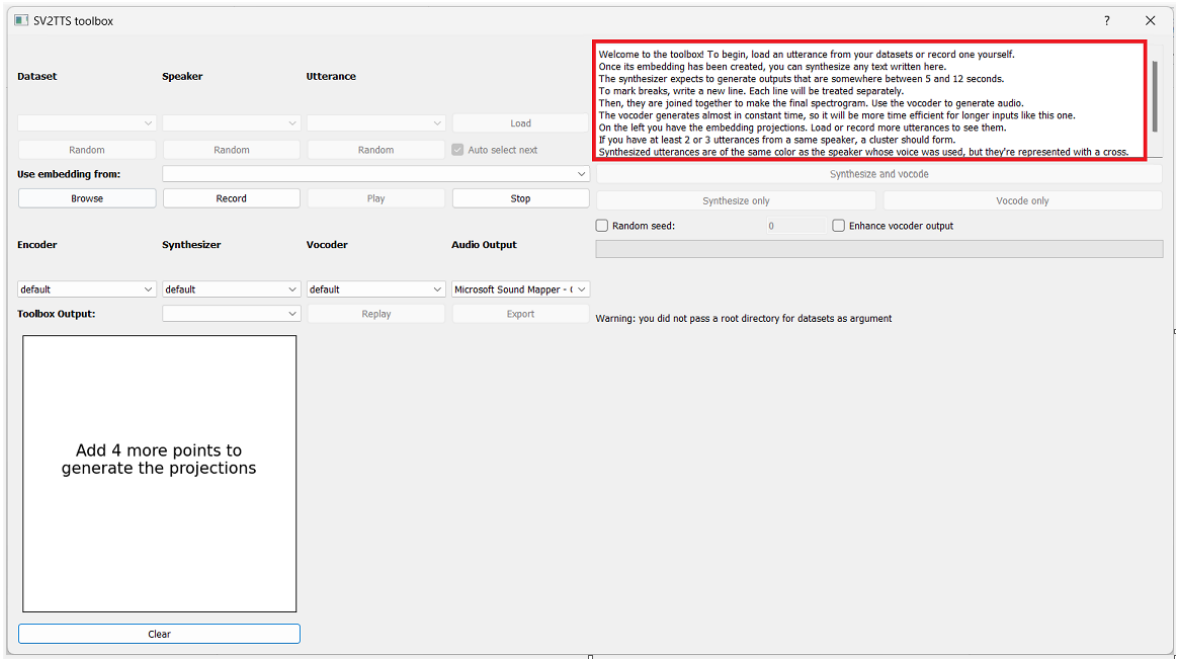
- Record: Dùng để ghi âm ngay lập tức (mặc định là 5 giây)



Hình 32. Giao diện kết quả (3)

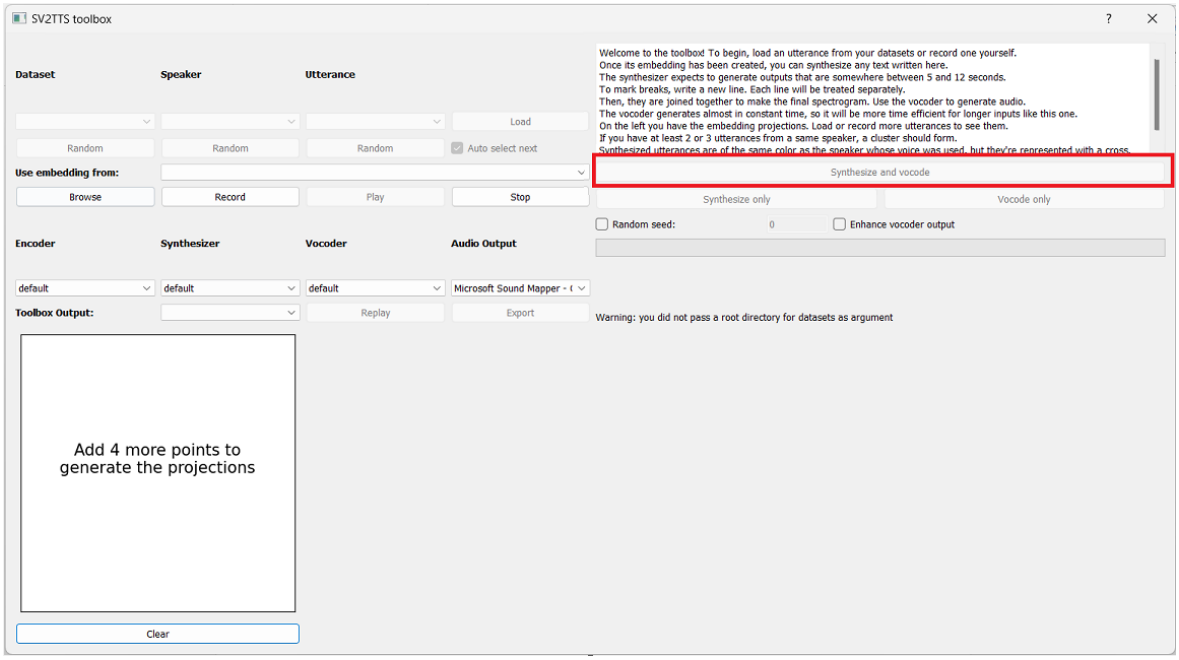
- Play: Dùng để phát lại tệp âm thành vừa tải lên hoặc vừa ghi âm bằng record.
- Stop: Dùng để dừng âm thanh đang phát.

Sau khi đã tải tệp âm thanh từ máy lên hoặc ghi âm từ record chúng ta cần điền đoạn văn bản ở phần khung màu đỏ ở hình bên dưới để sau khi hệ thống tổng hợp giọng nói hoàn tất sẽ dùng giọng được nhân bản để đọc đoạn văn bản đó.



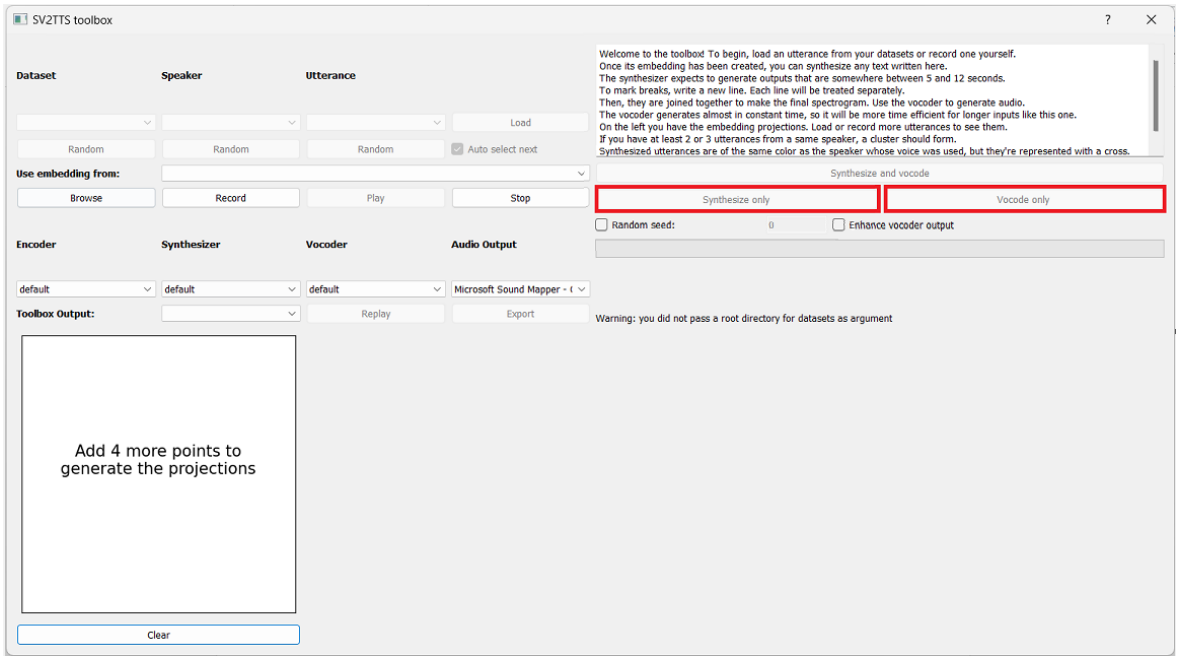
Hình 33. Giao diện kết quả (4)

Kế tiếp là chức năng tổng hợp và mã hoá (Synthesizer and vocoder) giọng nói để đọc văn bản. Lưu ý gộp này có thể làm cho một số máy tính không đủ mạnh bị treo.



Hình 34. Giao diện kết quả (5)

Do đó, chúng ta có thể thực hiện từng bước một thông qua 2 nút Chỉ tổng hợp giọng nói (Synthesizer only), sau khi tổng hợp xong có thể thực hiện tiếp chỉ giải mã giọng nói (Vocoder only) giải mã và đọc âm thanh.

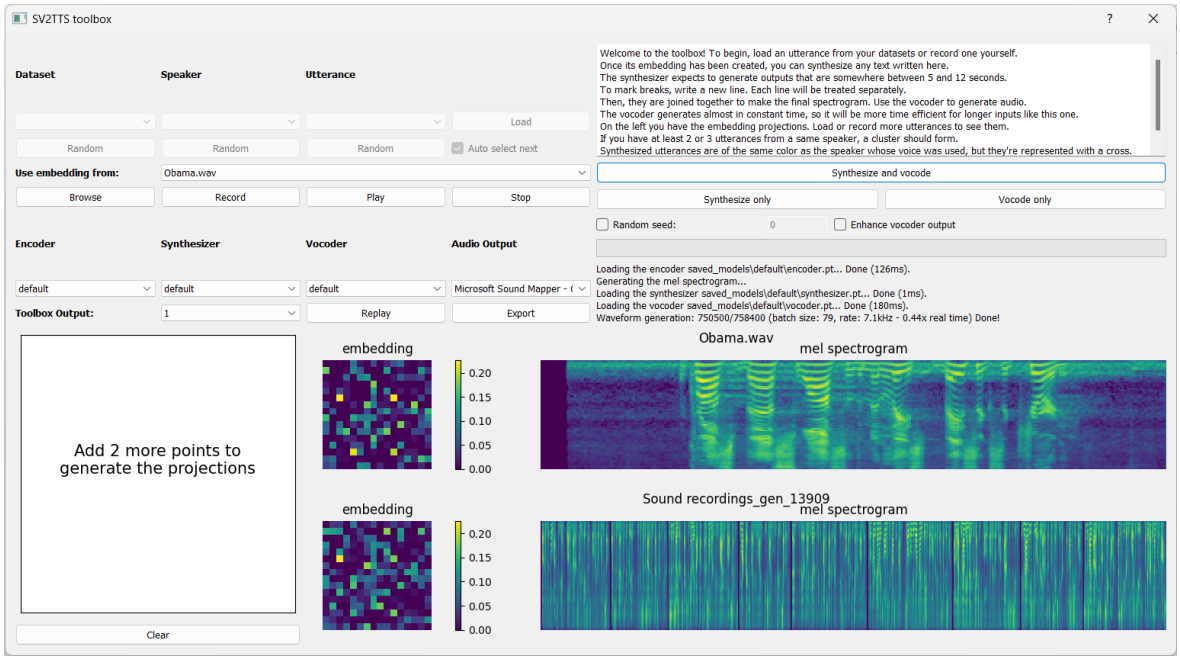


Hình 35. Giao diện kết quả (6)

3. Kết quả kiểm thử

Cài đặt thành công hệ thống có thể nhân bản được giọng nói từ một đoạn audio được thêm vào và sử dụng giọng nói đó để đọc được văn bản bất kì được thêm vào. Tuy nhiên độ chính xác và tính tự nhiên của giọng nói nhân bản còn chưa được cao, kèm theo đó là thời gian để thực hiện công việc nhân bản giọng nói này còn mất khá nhiều thời gian.

Đôi khi với một số giọng nói phần mềm không thể nhân bản được làm cho một số từ ngữ không thể đọc được bằng giọng nói hoặc thậm chí là không đọc được cả đoạn văn bản đầu vào.



Hình 36. Giao diện kết quả (7)

PHẦN KẾT LUẬN

1. Kết quả đạt được

- ✓ Xây dựng được ứng dụng nhân bản giọng nói từ một đoạn audio giọng nói đầu vào và dùng nó để đọc một đoạn bản bất kì.
- ✓ Đầu vào âm thanh có thể được tải lên từ máy tính hoặc được ghi âm trực tiếp ngay trên ứng dụng nhân bản giọng nói.
- ✓ Trực quan hoá dữ liệu âm thanh dưới dạng quang phổ mel.

2. Hướng phát triển

- ✓ Cải thiện độ chính xác của mô hình nhân bản giọng nói.
- ✓ Cải thiện thời gian tổng hợp giọng nói.
- ✓ Tăng độ tự nhiên của giọng nói sau khi được tổng hợp.
- ✓ Phát triển thành ứng dụng có thể chuyển đổi giọng nói của người dùng sang giọng nói của người khác ngay trong thời gian thực.

TÀI LIỆU THAM KHẢO

1. Ye Jia, Yu Zhang, Ron J. Weiss, Quan Wang, Jonathan Shen, Fei Ren, Zhifeng Chen, Patrick Nguyen, Ruoming Pang, Ignacio Lopez Moreno, Yonghui Wu. Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis
2. Georg Heigold, Ignacio Moreno, Samy Bengio, and Noam Shazeer. End-to-end text-dependent speaker verification. In Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on, pages 5115–5119. IEEE, 2016.
3. Ehsan Variani, Xin Lei, Erik McDermott, Ignacio Lopez Moreno, and Javier Gonzalez Dominguez. Deep neural networks for small footprint text-dependent speaker verification. In Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on, pages 4052–4056. IEEE, 2014.
4. Li Wan, Quan Wang, Alan Papir, and Ignacio Lopez Moreno. Generalized end-to-end loss for speaker verification. In Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2018.
5. Andrew Gibiansky, Sercan Arik, Gregory Diamos, John Miller, Kainan Peng, Wei Ping, Jonathan Raiman, and Yanqi Zhou. Deep Voice 2: Multi-speaker neural text-to-speech. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems 30, pages 2962–2970. Curran Associates, Inc., 2017.
6. Wei Ping, Kainan Peng, Andrew Gibiansky, Sercan O. Arik, Ajay Kannan, Sharan Narang, Jonathan Raiman, and John Miller. Deep Voice 3: 2000-speaker neural text-to-speech. In Proc. International Conference on Learning Representations (ICLR), 2018.
7. Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, RJ Skerry-Ryan, Rif A. Saurous, Yannis Agiomyrgiannakis, and Yonghui. Wu. Natural TTS synthesis by conditioning WaveNet on mel spectrogram predictions. In Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2018.
8. Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A generative model for raw audio. CoRR abs/1609.03499, 2016.

ĐƯỜNG LINK THAM KHẢO

[1] Đường link thư mục mô hình được đào tạo trước:
https://drive.google.com/drive/folders/1fU6umc5uQAVR2udZdHX-lDgXYzTyqG_j
[2] Đường link hướng dẫn sử dụng lớp cơ bản về mạng nơ ron của thư viện pytorch:
https://pytorch.org/docs/stable/generated/torch.nn.Module.html
[3] Kiến trúc tacotron 2:
https://github.com/NVIDIA/DeepLearningExamples/blob/master/PyTorch/SpeechSynthesis/Tacotron2/tacotron2/model.py
[4] Kiến trúc fatchord:
https://github.com/fatchord/WaveRNN/blob/master/models/fatchord_version.py
[5] Tài liệu sử dụng thư viện PyQt5:
https://www.riverbankcomputing.com/static/Docs/PyQt5/s