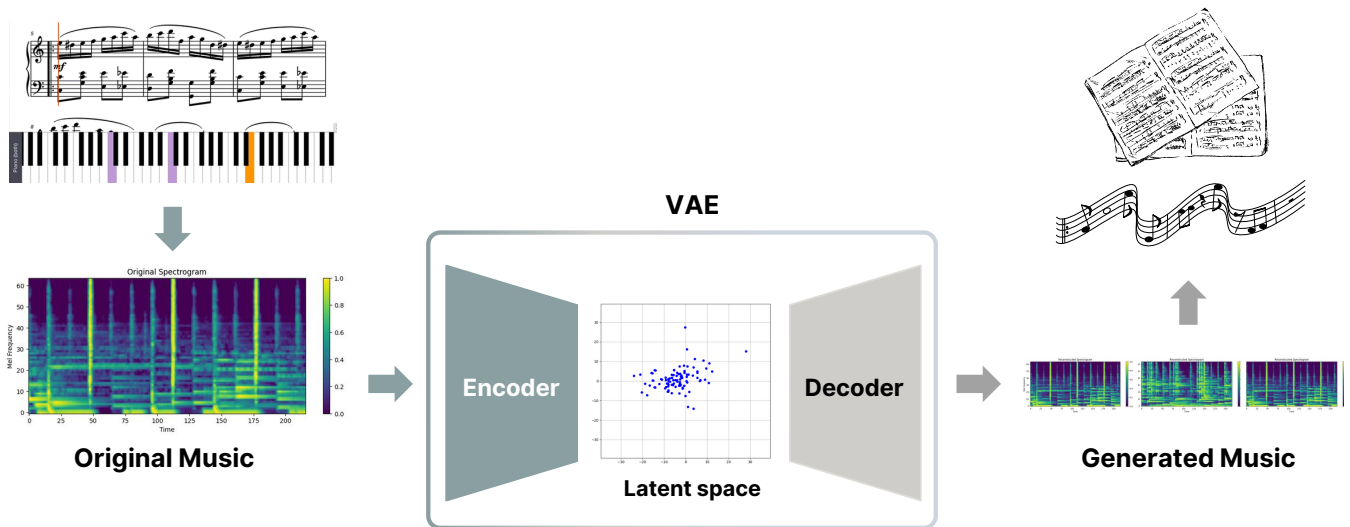


Project: Instrumental Music Generation

Dinh-Thang Duong, Yen-Linh Vu, Anh-Khoi Nguyen, Quang-Vinh Dinh

I. Giới thiệu

Music Generation là một bài toán thuộc lĩnh vực Xử lý âm thanh và các mô hình tạo sinh (generative models), liên quan đến việc xây dựng một hệ thống có khả năng tạo ra các đoạn nhạc mới dựa trên một đoạn nhạc mẫu (sound snippet) hoặc các tham số âm thanh nhất định. Trên thế giới, các dự án như [Magenta](#) của Google, [AIVA \(Artificial Intelligence Virtual Artist\)](#) hay [OpenAI Jukebox](#) đều là những ví dụ nổi tiếng trong lĩnh vực này. Mục tiêu của bài toán sinh nhạc không chỉ dừng lại ở việc sao chép phong cách của mẫu nhạc, mà còn hướng đến sự biến tấu sáng tạo, để có thể sinh ra các đoạn nhạc hoàn toàn mới và độc đáo.



Hình 1: Minh họa về bài toán sinh nhạc dựa trên một mẫu nhạc cho trước.

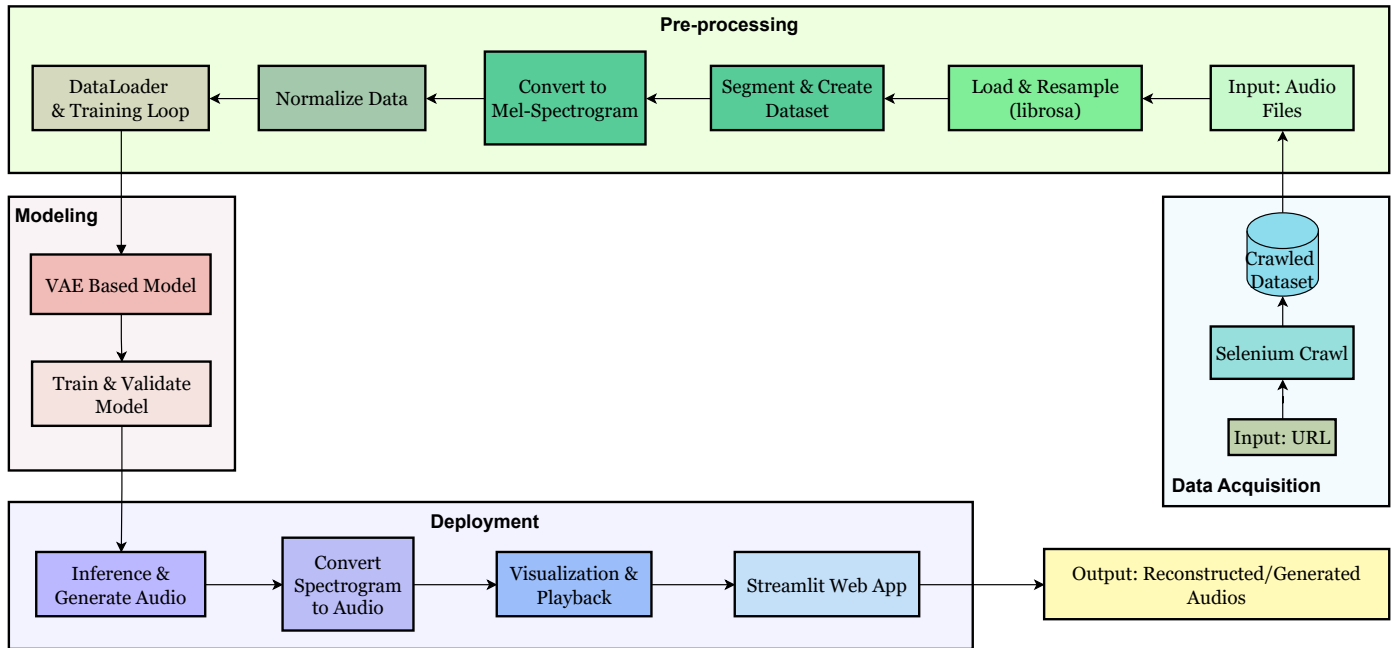
Trong project này, chúng ta sẽ cùng tìm hiểu phát triển một chương trình tạo sinh nhạc mới dựa trên một tín hiệu âm thanh gốc kèm theo một danh sách nhãn các thể loại nhạc. Tổng quan, Input và Output của bài toán như sau:

- **Input:** Một mẫu âm thanh và danh sách nhãn thể loại âm thanh đầu ra mong muốn.
- **Output:** Đoạn âm thanh mới được sinh ra, mang đặc trưng của đoạn âm thanh mẫu giao thoa với danh sách các nhãn thể loại đầu vào.

Mô hình được ứng dụng trong bài dựa trên kiến trúc mô hình *Variational Autoencoder (VAE)*. Theo đó, hệ thống sẽ học cách biểu diễn đặc trưng của âm thanh piano và nội dung các thể loại nhạc đi kèm thuộc các file âm thanh khác nhau vào trong một không gian tiềm ẩn (latent space), sau đó tận dụng không gian tiềm ẩn này để tạo ra những đoạn nhạc mới, mang đặc trưng tương đồng với mẫu nhạc gốc nhưng vẫn đảm bảo tính sáng tạo.

I.1. Project pipeline

Dựa trên các mô tả nội dung trên, ta có một pipeline tổng quát cho toàn bộ project được mô tả như ảnh sau:



Hình 2: Pipeline tổng quan của project.

Pipeline này có 4 giai đoạn chính:

- **Data Acquisition (Thu thập dữ liệu):** Dữ liệu âm thanh được thu thập từ web bằng Selenium.
- **Pre-processing (Tiền xử lý):** Dữ liệu được tải lên, xử lý và chuyển đổi sang định dạng Mel-Spectrogram.
- **Modeling (Huấn luyện mô hình):** Mô hình VAE được sử dụng để học cách biểu diễn âm thanh.
- **Deployment (Triển khai):** Mô hình được dùng để tạo dữ liệu âm thanh mới và hiển thị trên ứng dụng web.

I.1.1. Data Acquisition (Thu thập dữ liệu)

Dữ liệu âm thanh đầu vào được thu thập từ nhiều nguồn khác nhau. Đầu tiên, ta xác định các nguồn dữ liệu âm thanh phù hợp như các website lưu trữ nhạc miễn phí hoặc kho dữ liệu công khai. Sau đó, ta sử dụng Selenium để tự động thu thập dữ liệu từ các trang web, trích xuất danh sách các tệp âm thanh và siêu dữ liệu liên quan (tên bài hát, nghệ sĩ, thể loại, v.v.). Các đường dẫn tệp âm thanh này sau đó được sử dụng để tải xuống dữ liệu và lưu trữ chúng theo định dạng có tổ chức (ví dụ: thư mục theo thể loại, JSON chứa metadata, v.v.).

I.1.2. Pre-processing (Tiền xử lý)

Dữ liệu âm thanh sau khi thu thập được cần qua các bước tiền xử lý trước khi đưa vào mô hình. Đầu tiên, các tệp tin âm thanh được tải và chuẩn hóa tần số lấy mẫu bằng thư viện `librosa`. Tiếp theo, các đoạn âm thanh dài được chia nhỏ để tối ưu hóa việc huấn luyện. Sau đó, dữ liệu được chuyển đổi sang dạng Mel-Spectrogram bằng `librosa.feature.melspectrogram()`, giúp trích xuất đặc trưng quan trọng từ tín hiệu âm thanh. Cuối cùng, dữ liệu được chuẩn hóa bằng Min-Max Scaling hoặc Standardization để đảm bảo giá trị đầu vào có cùng thang đo, giúp mô hình học tốt hơn. Sau đó, dữ liệu được đưa vào `torch.utils.data.DataLoader` để tạo bộ tải dữ liệu và thiết lập vòng lặp huấn luyện với batch size phù hợp.

I.1.3. Modeling (Huấn luyện mô hình)

Mô hình chính được sử dụng là Biến Autoencoder (VAE), giúp học cách mã hóa và giải mã dữ liệu âm thanh. Mô hình gồm ba thành phần chính: **Encoder**, biến đổi đầu vào thành không gian tiềm ẩn; **Latent Space**, nơi mô hình học biểu diễn đặc trưng của âm thanh; và **Decoder**, tái tạo lại âm thanh từ không gian tiềm ẩn. Trong quá trình huấn luyện, ta tính toán hàm mất mát gồm *reconstruction loss* và *KL divergence loss*, sau đó kiểm tra mô hình trên tập validation để đánh giá hiệu suất.

I.1.4. Deployment (Triển khai)

Sau khi huấn luyện, mô hình được sử dụng để tạo ra các mẫu âm thanh mới từ không gian tiềm ẩn. Kết quả Mel-Spectrogram sau đó được chuyển đổi lại thành dạng sóng âm thanh bằng `librosa`. Để người dùng có thể tương tác, ta hiển thị Mel-Spectrogram của âm thanh gốc và âm thanh tái tạo, đồng thời cho phép phát lại trực tiếp trên ứng dụng. Một ứng dụng web dựa trên Streamlit được xây dựng để người dùng dễ dàng thử nghiệm và nghe các mẫu âm thanh đã tạo. Cuối cùng, các tệp âm thanh đầu ra được lưu trữ để đánh giá và cải thiện mô hình trong tương lai.

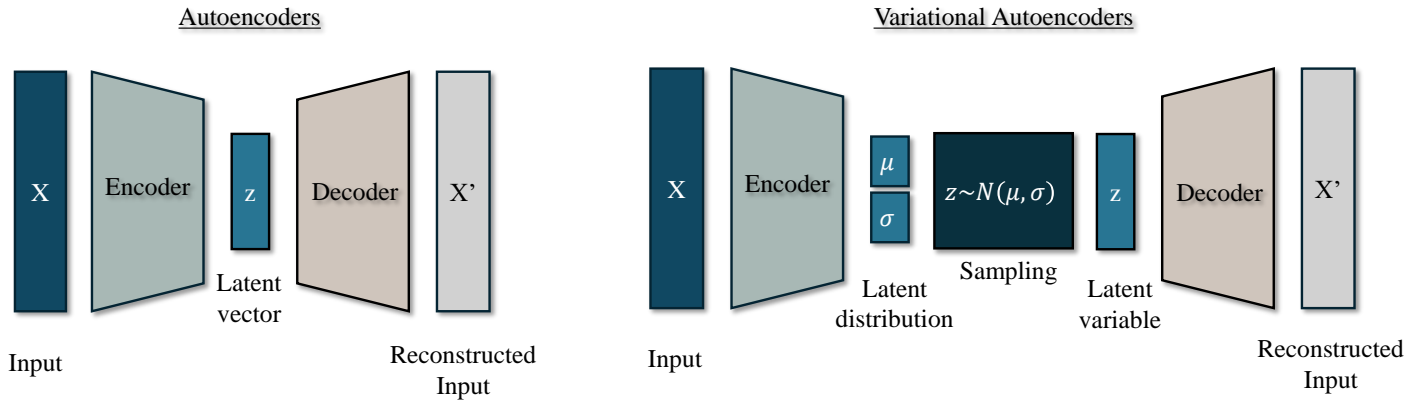
I.2. Tổng quan về VAE

Mục tiêu của pipeline là tạo ra một mô hình có khả năng không chỉ tái tạo mà còn sinh ra dữ liệu mới từ các đặc trưng trích xuất được. Để đạt được điều này, chúng ta hướng tới kiến trúc **Variational Autoencoder (VAE)**.

Khác với **Autoencoder (AE)** truyền thống, vốn chỉ ánh xạ dữ liệu đầu vào thành một điểm cụ thể trong không gian tiềm ẩn (*latent space*), VAE chuyển đổi đầu vào thành một phân phối

xác suất trong latent space. Điều này cho phép mô hình sinh ra dữ liệu mới bằng cách lấy mẫu từ phân phối xác suất đó, từ đó duy trì tính mạch lạc và khả năng tổng quát của thông tin.

I.2.1. Giới thiệu chung về Autoencoder (AE) và Variational Autoencoder (VAE)



Hình 3: Tổng quan về AE và VAE.

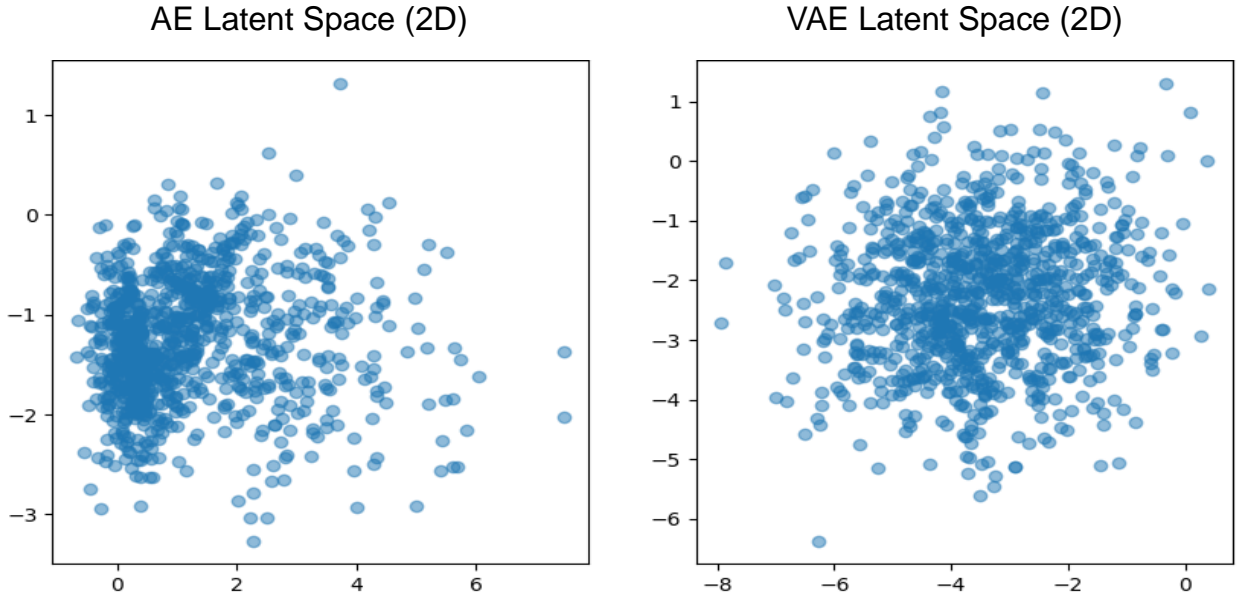
Autoencoder (AE): là một mạng nơ-ron sử dụng kiến trúc encoder-decoder để nén dữ liệu đầu vào vào một không gian thấp chiều (*low-dimensional latent space*) và tái tạo lại dữ liệu đó.

- **Encoder:** Biến đổi dữ liệu đầu vào thành một vector duy nhất trong latent space.
- **Decoder:** Tái tạo dữ liệu từ vector đó.

Variational Autoencoder (VAE): cải tiến AE bằng cách ánh xạ đầu vào thành một phân phối xác suất trong latent space (thường là Gaussian). Cụ thể:

- **Encoder** của VAE xuất ra hai tham số: **mean** μ và **variance** σ^2 , từ đó xây dựng phân phối $q(z | x)$.
- **Latent representation** không còn là một điểm duy nhất mà là một phân phối, cho phép sinh dữ liệu mới bằng cách lấy mẫu (sampling) từ đó.

Hình 4 dưới đây giúp ta thấy rõ nhược điểm của AE so với VAE. AE ánh xạ dữ liệu vào các điểm rời rạc trong không gian tiềm ẩn, dẫn đến sự phân bố không liên tục và có nhiều khoảng trống. Trong khi đó, VAE học được một phân phối xác suất có dạng Gaussian, giúp latent space liên tục và có cấu trúc hơn.



Hình 4: So sánh latent space (không gian tiềm ẩn) 2D của AE và VAE.

I.2.2. Các khái niệm cơ bản cần biết trong VAE

- **Latent Space (Không gian tiềm ẩn):** Là không gian trừu tượng nơi dữ liệu đầu vào được biểu diễn sau khi được mã hóa. Trong VAE, thay vì chỉ có một điểm, mỗi đầu vào được biểu diễn dưới dạng một phân phối (thường là Gaussian).
- **Latent Representation (Biểu diễn tiềm ẩn):** Là cách dữ liệu được biểu diễn trong latent space, thường dưới dạng các tham số như μ (mean) và σ^2 (variance) của phân phối Gaussian.
- **Tóm tắt cơ sở lý thuyết:**
 - $\mathbf{p}(\mathbf{x})$: *Data distribution* (phân phối dữ liệu), biểu diễn phân phối của toàn bộ tập dữ liệu đầu vào.
 - $\mathbf{p}(\mathbf{z})$: *Prior* (phân phối tiên nghiệm) của biến tiềm ẩn \mathbf{z} , phản ánh giả định ban đầu về không gian tiềm ẩn trước khi quan sát dữ liệu.
 - $\mathbf{p}(\mathbf{x}|\mathbf{z})$: *Likelihood* (xác suất tái tạo dữ liệu \mathbf{x} từ \mathbf{z}), mô tả cách dữ liệu được tạo ra từ không gian tiềm ẩn.
 - $\mathbf{q}(\mathbf{z}|\mathbf{x})$: *Posterior* (phân phối hậu nghiệm), đại diện cho phân phối của \mathbf{z} sau khi đã quan sát \mathbf{x} .

Trong thực tế, phân phối $\mathbf{p}(\mathbf{z})$ thường không được biết chính xác, khiến các tính toán trực tiếp trở nên khó khăn. Do đó, VAE giả định $\mathbf{p}(\mathbf{z})$ tuân theo một phân phối chuẩn $\mathcal{N}(0, I)$ để đơn giản hóa mô hình.

Tuy nhiên, việc tính toán $\mathbf{p}(\mathbf{z}|\mathbf{x})$ trực tiếp là không khả thi. Thay vào đó, VAE sử dụng $\mathbf{q}(\mathbf{z}|\mathbf{x})$ để xấp xỉ phân phối này. Mục tiêu của mô hình là tối ưu hóa hàm *ELBO* (Evidence Lower Bound)

để làm cho $q(\mathbf{z}|\mathbf{x})$ gần nhất với $p(\mathbf{z}|\mathbf{x})$.

Lý thuyết VAE dựa trên định lý Bayes, trong đó $q(\mathbf{z}|\mathbf{x})$ được sử dụng để xấp xỉ $p(\mathbf{z}|\mathbf{x})$. Để tìm hiểu sâu hơn có thể đọc thêm tại [đây](#).

I.2.3. Loss function của VAE

Hàm mất mát của VAE được gọi là ELBO (Evidence Lower Bound) bao gồm hai thành phần chính:

$$L(x) = \mathbb{E}_{q(z|x)}[\log p(x|z)] - D_{KL}(q(z|x) \parallel p(z))$$

Trong đó:

- $\mathbb{E}_{q(z|x)}[\log p(x|z)]$: **Reconstruction Loss** (thường dùng MSE hoặc BCE), đo lường khả năng tái tạo dữ liệu từ latent space.
- $D_{KL}(q(z|x) \parallel p(z))$: **KL Divergence**, đo khoảng cách giữa phân phối xấp xỉ $q(z|x)$ và phân phối chuẩn $p(z)$ (thường là $\mathcal{N}(0, I)$).

Mục tiêu của VAE là tối ưu hóa ELBO (Evidence Lower Bound):

$$\log p(x) \geq \mathbb{E}_{q(z|x)}[\log p(x|z)] - D_{KL}(q(z|x) \parallel p(z))$$

Bằng cách tối đa hóa ELBO, ta đồng thời giảm thiểu lỗi tái tạo và đảm bảo phân phối tiềm ẩn có cấu trúc hợp lý.

I.2.4. Reparameterization Trick

Trong VAE, ta cần tối ưu hóa $q(z|x)$, tức là phải lấy mẫu $z \sim \mathcal{N}(\mu, \sigma^2)$. Tuy nhiên, phép lấy mẫu là một quá trình không khả vi, gây khó khăn cho việc lan truyền gradient trong quá trình huấn luyện mô hình.

Để giải quyết vấn đề này, VAE sử dụng **Reparameterization Trick**, giúp biến đổi phép lấy mẫu thành một phép toán khả vi:

$$z = \mu + \sigma \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0, 1)$$

Cách tiếp cận này cho phép tính đạo hàm qua các tham số μ và σ , giúp huấn luyện mô hình bằng các thuật toán tối ưu hóa dựa trên gradient (như Adam).

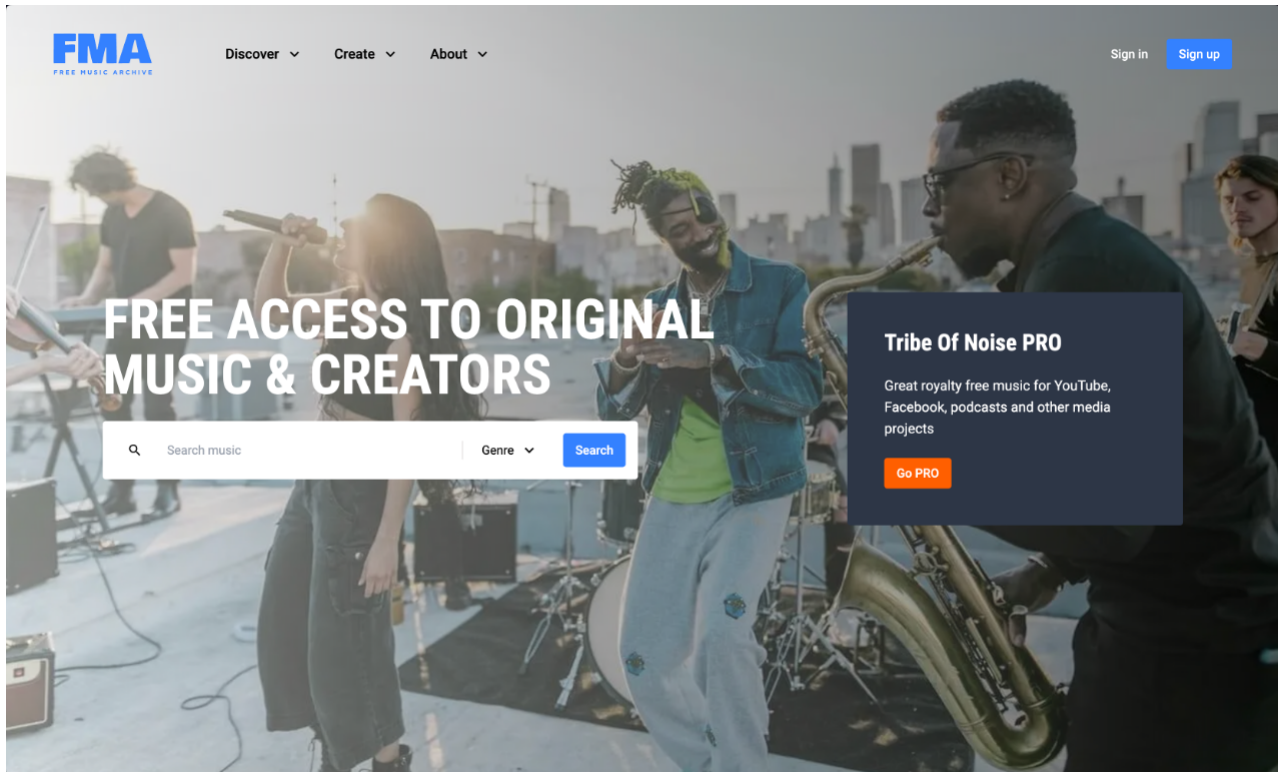
Variational Autoencoder (VAE) là một bước tiến vượt bậc so với Autoencoder truyền thống, nhờ vào việc sử dụng latent space có cấu trúc xác suất. Điều này không chỉ giúp tái tạo dữ liệu đầu vào mà còn cho phép mô hình sinh ra các mẫu dữ liệu mới, mở ra nhiều ứng dụng trong tạo ảnh, tổng hợp văn bản, và nhiều lĩnh vực khác.

II. Cài đặt chương trình

Trong phần này, chúng ta sẽ tìm hiểu về quá trình xây dựng toàn bộ chương trình Music Generation sử dụng VAE. Trong đó, bao gồm hai phần nội dung lớn là thu thập bộ dữ liệu để huấn luyện mô hình và xây dựng mô hình VAE.

II.1. Thu thập bộ dữ liệu

Tại project này, chúng ta giả định trong trường hợp chưa có một bộ dữ liệu có sẵn. Vì vậy, việc thu thập dữ liệu cần phải được thực hiện. Dựa vào nội dung của project, một lượng các file âm thanh có tiếng piano cần được thu thập để có thể huấn luyện mô hình VAE. Có rất nhiều nguồn cũng như cách thức để thu thập dữ liệu, trong đó, nguồn từ internet là một trong những nguồn thu thập dễ tiếp cận nhất. Đối với phạm vi của project này, chúng ta sẽ sử dụng thư viện Selenium để thực hiện thu thập dữ liệu trên một trang web chuyên lưu trữ các file âm nhạc.



Hình 5: Trang chủ của website freemusicarchive.org.

Lưu ý: Các bạn có thể tải trực tiếp bộ dữ liệu đã được thu thập sẵn ở phần [IV](#). và bỏ qua phần này.

II.1.1. Cài đặt Chrome Driver và Selenium

Chúng ta sẽ thực hiện việc thu thập dữ liệu bằng Selenium trên Colab. Để sử dụng Selenium, chúng ta cần thực thi một số lệnh cài đặt phức tạp sau đây. Tổng quan, Colab sẽ cần được cài

đặt Chrome Driver và Selenium. Đoạn mã bash có nội dung như sau:

```
1 %%shell
2 # Ubuntu no longer distributes chromium-browser outside of snap
3 #
4 # Proposed solution: https://askubuntu.com/questions/1204571/how-to-install-
   chromium-without-snap
5
6 # Add debian buster
7 cat > /etc/apt/sources.list.d/debian.list << "EOF"
8 deb [arch=amd64 signed-by=/usr/share/keyrings/debian-buster.gpg] http://deb.
   debian.org/debian buster main
9 deb [arch=amd64 signed-by=/usr/share/keyrings/debian-buster-updates.gpg] http
   ://deb.debian.org/debian buster-updates
   main
10 deb [arch=amd64 signed-by=/usr/share/keyrings/debian-security-buster.gpg] http
   ://deb.debian.org/debian-security
   buster/updates main
11 EOF
12
13 # Add keys
14 apt-key adv --keyserver keyserver.ubuntu.com --recv-keys DCC9EFBF77E11517
15 apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 648ACFD622F3D138
16 apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 112695A0E562B32A
17
18 apt-key export 77E11517 | gpg --dearmor -o /usr/share/keyrings/debian-buster.
   gpg
19 apt-key export 22F3D138 | gpg --dearmor -o /usr/share/keyrings/debian-buster-
   updates.gpg
20 apt-key export E562B32A | gpg --dearmor -o /usr/share/keyrings/debian-
   security-buster.gpg
21
22 # Prefer debian repo for chromium* packages only
23 # Note the double-blank lines between entries
24 cat > /etc/apt/preferences.d/chromium.pref << "EOF"
25 Package: *
26 Pin: release a=eoan
27 Pin-Priority: 500
28
29
30 Package: *
31 Pin: origin "deb.debian.org"
32 Pin-Priority: 300
33
34
35 Package: chromium*
36 Pin: origin "deb.debian.org"
37 Pin-Priority: 700
38 EOF
39
40 # Install chromium and chromium-driver
41 apt-get update
42 apt-get install chromium chromium-driver
43
44 # Install selenium
45 pip install selenium
```


II.1.2. Import các thư viện cần thiết

Sau khi tải và cài đặt thành công Selenium và Chrome Driver, ta tiến hành import các thư viện cần thiết để sử dụng được đoạn code thu thập dữ liệu như sau:

```

1 import os
2 import requests
3 import time
4 import json
5
6 from tqdm import tqdm
7 from selenium import webdriver
8 from selenium.webdriver.chrome.service import Service
9 from selenium.webdriver.common.by import By
10 from selenium.webdriver.support.ui import WebDriverWait
11 from selenium.webdriver.support import expected_conditions as EC

```

II.1.3. Khởi tạo trình duyệt

Bắt đầu chương trình, ta cần khởi tạo một Chrome Driver để có thể truy cập vào các trang web thông qua driver này. Một số cài đặt `chrome_options` được thêm vào trong đoạn code dưới đây liên quan đến driver để phù hợp với môi trường Colab:

```

1 WEBDRIVER_DELAY_TIME_INT = 10
2 TIMEOUT_INT = 10
3 service = Service(executable_path=r"/usr/bin/chromedriver")
4 chrome_options = webdriver.ChromeOptions()
5 chrome_options.add_argument("--headless")
6 chrome_options.add_argument("--no-sandbox")
7 chrome_options.add_argument("--disable-dev-shm-usage")
8 chrome_options.add_argument("window-size=1920x1080")
9 chrome_options.headless = True
10 driver = webdriver.Chrome(service=service, options=chrome_options)
11 driver.implicitly_wait(TIMEOUT_INT)
12 wait = WebDriverWait(driver, WEBDRIVER_DELAY_TIME_INT)

```

II.1.4. Xây dựng hàm tách thông tin từ danh sách nhạc

Ở trang web nhạc này, mục tiêu của chúng ta là, với một bảng danh sách các nhạc của một thể loại (genres) nào đó, xây dựng một chương trình trích xuất và duyệt qua được từng hàng nội dung âm thanh một. Nguyên nhân là vì,

```

1 def extract_audio_links_from_menu(menu_url, driver):
2     driver.get(menu_url)
3     container = wait.until(EC.presence_of_element_located(
4         (By.CSS_SELECTOR, "div.w-full.flex.flex-col.gap-3.pt-3")
5     ))
6     play_items = container.find_elements(By.CSS_SELECTOR, "div.play-item")
7     links = []
8     for item in play_items:
9         try:
10             a_tag = item.find_element(By.CSS_SELECTOR, ".ptxt-track a")
11             link = a_tag.get_attribute("href")

```

```

12         links.append(link)
13     except Exception:
14         continue
15     return links

```

The screenshot shows a music player interface. On the left, a table lists tracks under the 'Piano' genre. The first track, 'Awaken with the Moon' by Honey Shade, is highlighted with a red box and a green circle labeled '1'. To the right, a detailed view of this track is shown, including its album cover, title, artist, and a 'Track info' section. A red box and a green circle labeled '2' highlight the track title and artist. Below the track info, a code snippet is shown with a green circle labeled '3' pointing to a URL. A green circle labeled '4' points to a URL in the code snippet.

Track	Album	Genres	Duration
Awaken with the Moon by Honey Shade	Ealing to Lake O	Pop, Experimental Pop, Piano	02:28
Gounod / Liszt - Hymne a Sainte Cecile - CG 557 / S. 491 by Gregor Quendel	Piano Classics Collect...	Classical, Soundtrack, Piano	08:34
Creepy Piano 1 by HoltznacCO	Background Music	Soundtrack, Piano, Halloween	02:17
Creepy Piano 2 by HoltznacCO	Background Music	Soundtrack, Piano, Halloween	02:22
Creepy Piano 3 by HoltznacCO	Background Music	Soundtrack, Piano, Halloween	01:18
Spring Morning by Maarten Schellekens	Free Neoclassical Mu...	Piano, Contemporary Classical, Instrumen	02:02
Bluesy by Lite Saturation	Piano Blues Collection	Blues, Soundtrack, Piano	01:58

Track info

Description: Free to use in any project even commercial WITH attribution. Have Fun!

Album: Ealing to Lake O

Genres: Pop, Experimental Pop, Piano

Instrumental: Yes

Explicit: Radio-Safe

AI generated?: No

Released: Mar 19, 2025

Plays: 59

Downloads: 0

Favorites: 0

Code snippet:

```

<div class="flex items-center gap-4 px-4 py-2 bg-gray-100
ht py-4 rounded mx-auto w-full gcol gid-electronic tid-24
3226 play-item" data-track-info{"id":243226,"handle":"a
waken-with-the-moon","url":"https://freemusicarchive.org
g/music/honey-shade/ealing-to-lake-o/awaken-with-the-moon/","title":"Awaken with the Moon","artistName":"Honey
Shade","artistUrl":"https://freemusicarchive.org/music/honey-shade/","albumTitle":"Ealing to Lake O","playb
ackUrl":"https://freemusicarchive.org/track/awaken-wi
n-the-moon/stream/","downloadUrl":"https://freemusi
chive.org/track/awaken-with-the-moon/download/","fl
oId":"TzorbJ6BFHzuQo6O2mMQgvnJeyvW15FKRbW4a.mp3"}">
<div class="flex items-center gap-4 px-4 py-2 bg-gray-100
ht py-4 rounded mx-auto w-full gcol gid-electronic tid-24
3226 play-item" data-track-info{"id":243226,"handle":"a
waken-with-the-moon","url":"https://freemusicarchive.org
g/music/honey-shade/ealing-to-lake-o/awaken-with-the-moon/","title":"Awaken with the Moon","artistName":"Honey
Shade","artistUrl":"https://freemusicarchive.org/music/honey-shade/","albumTitle":"Ealing to Lake O","playb
ackUrl":"https://freemusicarchive.org/track/awaken-wi
n-the-moon/stream/","downloadUrl":"https://freemusi
chive.org/track/awaken-with-the-moon/download/","fl
oId":"TzorbJ6BFHzuQo6O2mMQgvnJeyvW15FKRbW4a.mp3"}">

```

Hình 6: Tổng quan các bước để tìm và trích xuất được nội dung file âm thanh có thể tải về máy tự động. Các file âm thanh này thuộc từng hàng nội dung bài nhạc trên bảng menu.

II.1.5. Xây dựng hàm tải file âm thanh

Ta xây dựng hàm tải file với đầu vào là một đường dẫn cho trước. Về mặt kỹ thuật, hàm này hoàn toàn có thể sử dụng để tải bất cứ file dữ liệu nào trên mạng, song trong trường hợp này chúng ta chỉ dùng để tải file âm thanh. Nội dung hàm này như sau:

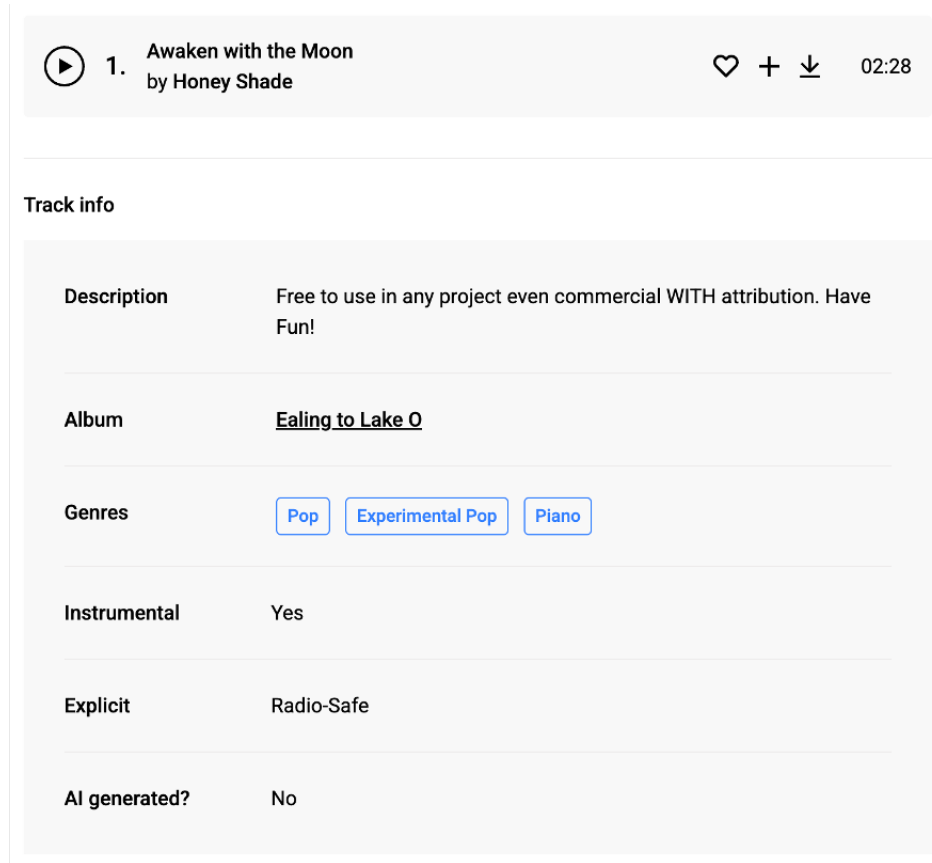
```

1 def download_audio_file(file_url, filepath):
2     response = requests.get(file_url, stream=True)
3     if response.status_code == 200:
4         with open(filepath, "wb") as f:
5             for chunk in response.iter_content(chunk_size=8192):
6                 f.write(chunk)
7     else:
8         print(f"Error downloading file from {file_url}; status: {response.status_code}")

```

II.1.6. Xây dựng các hàm trích xuất nội dung và thông tin trang nhạc

Quan sát các thành phần/nội dung được hiển thị trên trang web của bài nhạc, có thể nhận thấy bên cạnh file âm thanh còn có những thông tin hữu ích mà ta hoàn toàn có thể kéo về đồng thời. Các thông tin đi kèm này hoàn toàn có thể được sử dụng cho các mục đích, ý tưởng khác nhằm mở rộng bài project sau này.



1. **Awaken with the Moon**
by Honey Shade

02:28

Track info

Description	Free to use in any project even commercial WITH attribution. Have Fun!
Album	<u>Ealing to Lake O</u>
Genres	Pop Experimental Pop Piano
Instrumental	Yes
Explicit	Radio-Safe
AI generated?	No

Hình 7: Ví dụ minh họa về một bảng thông tin được cung cấp sẵn trên trang web cho một bài nhạc.

Dựa trên những gì được hiển thị trong hình 7, ta sẽ xây dựng các hàm riêng biệt để trích xuất giá trị các trường thông tin. Các hàm đều nhận đầu vào là chrome driver đã được truy cập sẵn vào website của bài nhạc. Ý nghĩa của các đoạn code được điều chỉnh phông theo cấu trúc HTML hiện thời của trang web để có thể trích được thông tin tương ứng.

1. Hàm tách nội dung đường dẫn chứa file âm thanh

```

1 def extract_track_info(driver):
2     audio_div = WebDriverWait(driver, 15).until(
3         EC.presence_of_element_located((By.CSS_SELECTOR, "div[data-track-info]
4         )))
5     return json.loads(audio_div.get_attribute("data-track-info"))

```

2. Hàm tách các nhãn thể loại nhạc

```

1 def extract_genres(driver):
2     try:
3         genre_elem = driver.find_element(By.CSS_SELECTOR, "span.md\\:col-span-
4         6.flex.flex-wrap.gap-3")
5         return [a.text.strip() for a in genre_elem.find_elements(By.TAG_NAME,
6         "a") if a.text.strip()]
7     except Exception:
8         return []

```

3. Hàm tách thời lượng nhạc

```

1 def extract_duration(driver):
2     try:
3         duration_elem = driver.find_element(By.CSS_SELECTOR, "span.w-12.ml-
                                auto.md\\:ml-0.col-span-2.inline-flex.
                                justify-end.items-center")
4         return duration_elem.text.strip()
5     except Exception:
6         return ""

```

4. Hàm tách một số thông tin khác

Hiện tại hàm này sẽ tách thông tin cho ta biết rằng liệu bài nhạc có phải là sản phẩm từ mô hình AI hay không? Bài nhạc này có lời hay không?

```

1 def extract_extra_info(driver):
2     instrumental = "No"
3     ai_generated = "No"
4     try:
5         info_container = driver.find_element(By.CSS_SELECTOR, "div.px-8.py-2.
                                bg-gray-light.flex.flex-col.divide-y.
                                divide-gray")
6         info_divs = info_container.find_elements(By.CSS_SELECTOR, "div.grid.
                                grid-cols-1.md\\:grid-cons-8.py-6")
7         for div in info_divs:
8             label = div.find_element(By.CSS_SELECTOR, "span.font-\\[500\\].md
                                \\:col-span-2").text.strip()
9             value = div.find_element(By.CSS_SELECTOR, "span.md\\:col-span-6").
                                text.strip()
10            if "Instrumental" in label:
11                instrumental = value
12            if "AI generated?" in label:
13                ai_generated = value
14    except Exception:
15        pass
16    return instrumental, ai_generated

```

Cuối cùng, tổng hợp tất cả các hàm trên với từng vai trò riêng biệt, ta tạo một hàm nhận đầu vào là một đường dẫn đến website của một bài nhạc bất kì. Hàm này sau đó sử dụng driver để truy cập vào trang web và lần lượt gọi các hàm đã định nghĩa ở trên để trích xuất thông tin, cuối cùng tổng hợp lại thành một dictionary để lưu xuống máy dưới dạng file .json cũng như tải file âm thanh tương ứng.

```

1 def process_audio_page(audio_url, driver, index):
2     driver.get(audio_url)
3
4     track_info = extract_track_info(driver)
5     file_url = track_info.get("fileUrl", "")
6     audio_name = track_info.get("title", "").strip()
7     author = track_info.get("artistName", "").strip()
8
9     genres = extract_genres(driver)
10    duration = extract_duration(driver)
11    instrumental, ai_generated = extract_extra_info(driver)
12
13    metadata = {

```

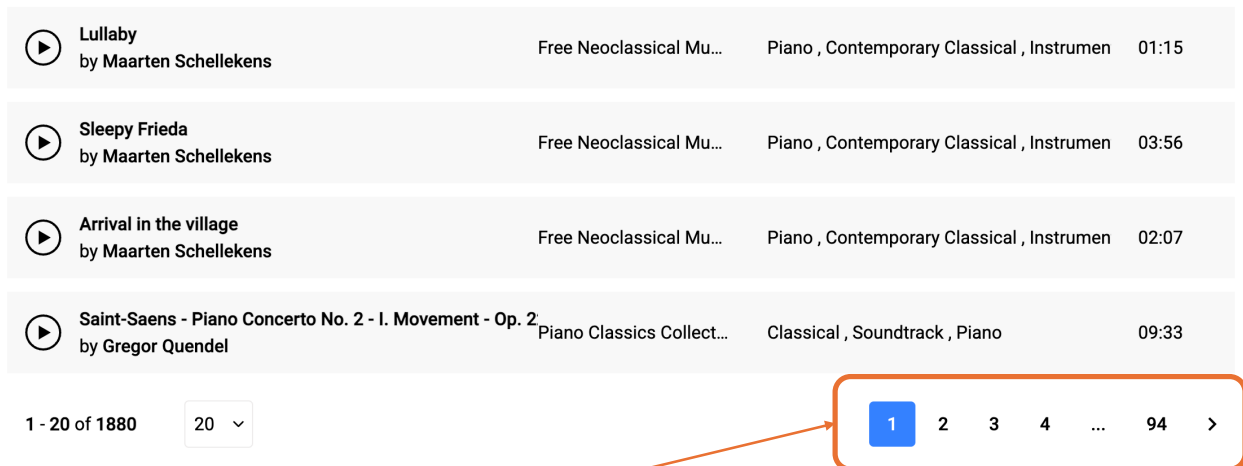
```

14     "audioName": audio_name,
15     "author": author,
16     "genres": genres,
17     "instrumental": instrumental,
18     "ai_generated": ai_generated,
19     "duration": duration,
20     "audio_url": audio_url
21 }
22
23 audio_filename = f"audio_{index:04d}.mp3"
24 meta_filename = f"audio_{index:04d}.json"
25 audio_filepath = os.path.join("crawled_data", "audio", audio_filename)
26 meta_filepath = os.path.join("crawled_data", meta_filename)
27
28 download_audio_file(file_url, audio_filepath)
29
30 with open(meta_filepath, "w", encoding="utf-8") as f:
31     json.dump(metadata, f, ensure_ascii=False, indent=4)
32
33 return metadata

```

II.1.7. Xây dựng hàm duyệt từng danh sách nhạc

Có thể thấy ở bảng menu danh sách nhạc, ta có thể chuyển sang nhiều menu (page) khác sau đó. Dựa theo danh sách ở thể loại Piano, có đến 94 trang menu với các bài nhạc khác nhau. Chính vì vậy, chúng ta cần một hàm có thể tự động chuyển sang các page khác để có thể lấy nhiều bài nhạc hơn. Dựa trên thông tin có ở trang web, ta xây dựng hàm này như sau:



<https://freemusicarchive.org/genre/piano/?page=1>

Hình 8: Minh họa các thông tin giúp ta có thể duyệt qua các page danh sách nhạc tự động.

```

1 def loop_over_menu_pages(base_url, total_pages, driver):
2     all_links = []
3     for page in tqdm(range(1, total_pages + 1), desc="Extracting Links", unit=
                        "page"):

```

```

4     page_url = f"{base_url}?page={page}"
5     try:
6         links = extract_audio_links_from_menu(page_url, driver)
7         all_links.extend(links)
8     except Exception as e:
9         print(f"Error on page {page}: {e}")
10    return all_links

```

II.1.8. Thực thi quá trình thu thập dữ liệu

Cuối cùng, tổng hợp toàn bộ các code phía trên, ta triển khai việc thu thập dữ liệu âm thanh trên trang web freemusicarchive.org. Trong nội dung project này, ta chỉ xem xét thu thập các bài nhạc có thể loại là **Piano**. Song, các bạn hoàn toàn có thể điều chỉnh để thu thập các thể loại nhạc khác. Code triển khai như sau:

```

1  os.makedirs("crawled_data", exist_ok=True)
2  os.makedirs(os.path.join("crawled_data", "audio"), exist_ok=True)
3
4  base_url = "https://freemusicarchive.org/genre/piano/"
5  total_pages = 10
6  sample_idx = 1
7  audio_links = loop_over_menu_pages(base_url, total_pages, driver)
8  print(f"Total audio links extracted: {len(audio_links)}")
9
10 for audio_url in tqdm(audio_links, desc="Downloading Audio and Metadata", unit
                        ="audio"):
11     try:
12         process_audio_page(audio_url, driver, sample_idx)
13         sample_idx += 1
14     except:
15         print(f'Error at: {audio_url}')
16         continue
17     time.sleep(0.5)
18
19 driver.quit()

```

Quá trình thu thập sẽ diễn ra nhanh hay chậm tùy thuộc vào số lượng trang page mà chúng ta muốn tải về. Khi kết thúc thực thi đoạn mã trên, ta nhận được một thư mục `crawled_data` có cấu trúc như sau:

```

crawled_data
├── audio
│   ├── audio_0001.mp3
│   ├── audio_0002.mp3
│   └── ...
├── audio_0001.json
├── audio_0002.json
└── ...

```

Như vậy, quá trình thu thập dữ liệu đã hoàn tất, và chúng ta đã có được một bộ dữ liệu âm thanh Piano để tiến hành huấn luyện mô hình trong phần sau.

II.2. Xây dựng mô hình

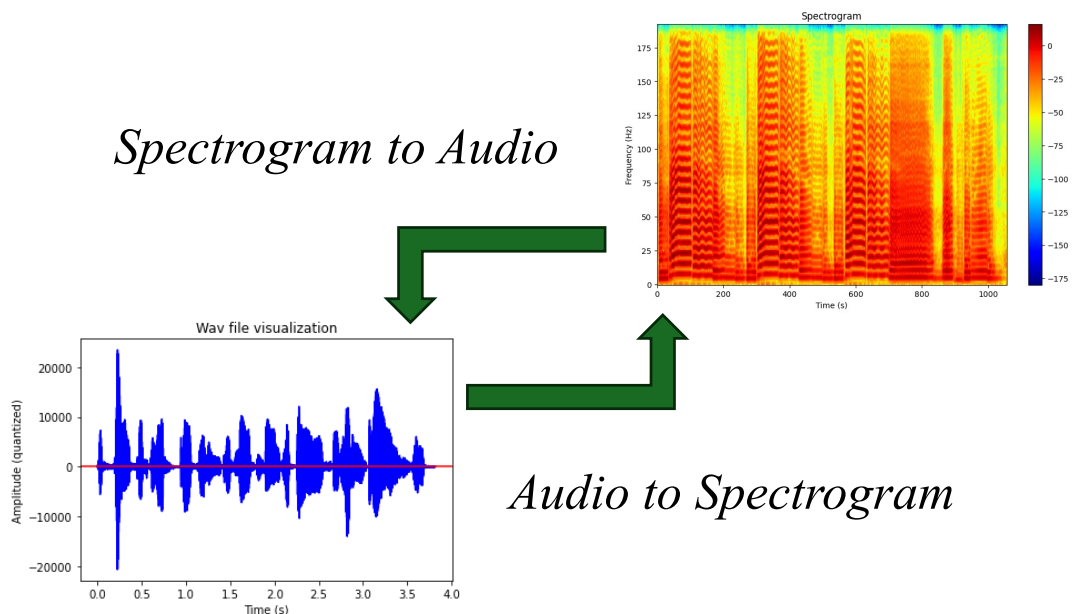
II.2.1. Import các thư viện cần thiết

Trước tiên, chúng ta cần tải các thư viện không có sẵn trên colab với phiên bản phù hợp:

```
1 !pip install numba==0.61.0 torchaudio==2.6.0 librosa==0.10.2.post1
```

Sau đó, thực hiện import các thư viện cần thiết cho chương trình project này:

```
1 import os
2 import torch
3 import torch.nn as nn
4 import torch.optim as optim
5 import torchaudio
6 import torchaudio.transforms as T
7 import numpy as np
8 import librosa
9 import librosa.display
10 import IPython.display as ipd
11 import torch.nn.functional as F
12 import matplotlib.pyplot as plt
13 import json
14
15 from sklearn.preprocessing import MinMaxScaler
16 from torch.utils.data import Dataset, DataLoader
17 from IPython.display import Audio
18 from tqdm import tqdm
19
20 device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```



Hình 9: Minh họa về việc chuyển đổi tín hiệu âm thanh sang ảnh mel spectrogram và ngược lại.

II.2.2. Tải bộ dữ liệu

Ta tải bộ dữ liệu đã được thu thập từ phần trước và giải nén để sử dụng:

```
1 !gdown 1u2WzsWUlyZbbPDfXAWXuLRMTwFkT21wa
2 !unzip -q crawled_piano_audio_40_pages.zip -d piano_audio_files_40_pages
3 os.remove("crawled_piano_audio_40_pages.zip")
```

II.2.3. Tiền xử lý bộ dữ liệu âm thanh

Để mô hình có thể xử lý và hiểu được các đặc trưng từ dữ liệu âm thanh một cách hiệu quả, chúng ta không trực tiếp sử dụng dữ liệu đọc được từ file âm thanh gốc (tín hiệu âm thanh miền thời gian) mà sẽ thực hiện một vài bước chuyển đổi để chuyển thành dạng biểu diễn mel spectrogram của âm thanh (hình 9). Đây là một kiểu biểu diễn truyền thống trong xử lý tín hiệu âm thanh vì nó mang lại nhiều thông tin có lợi hơn cho việc học của mô hình. Theo đó, chúng ta sẽ khai báo một số hàm như sau cho quá trình tiền xử lý toàn bộ dữ liệu:

- Đọc tệp json và lấy thông tin thể loại `load_and_get_genres()`.
- Đọc và lấy mẫu âm thanh `load_and_resample_audio()`.
- Chuyển dữ liệu âm thanh gốc sang dạng mel spectrogram `audio_to_melspec()` và ngược lại `melspec_to_audio()`.
- Hiển thị biểu đồ mel spectrogram `show_spectrogram()`.
- Chuẩn hoá các giá trị về miền (0, 1) để phù hợp khi đưa vào mô hình `normalize_melspec()` và khử chuẩn hoá để đưa về giá trị trước khi chuẩn hoá `denormalize_melspec()`.
- Hiển thị âm thanh với `display_audio_files()`.

```
1 def load_and_get_genres(json_path):
2     with open(json_path, "r") as f:
3         data = json.load(f)
4
5     return data.get('genres', [])
6
7 def load_and_resample_audio(file_path, target_sr=22050):
8     audio, sr = librosa.load(file_path, sr=None)
9     if sr != target_sr:
10         audio = librosa.resample(audio, orig_sr=sr, target_sr=target_sr)
11
12     return audio, target_sr
13
14 def audio_to_melspec(audio, sr, n_mels, n_fft=2048, hop_length=512, to_db=
15                                     False):
16     spec = librosa.feature.melspectrogram(
17         y=audio,
18         sr=sr,
19         n_fft=n_fft,
20         hop_length=hop_length,
21         win_length=None,
```

```

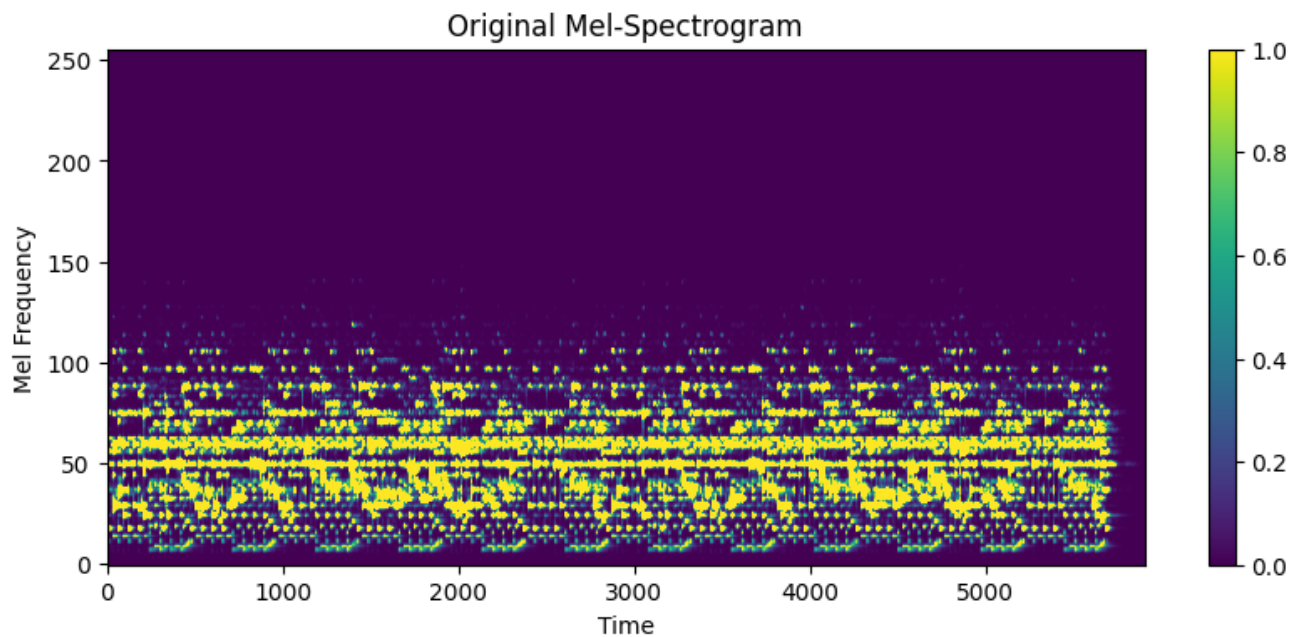
21         window="hann",
22         center=True,
23         pad_mode="reflect",
24         power=2.0,
25         n_mels=n_mels
26     )
27     if to_db:
28         spec = librosa.power_to_db(spec, ref=np.max)
29
30     return spec
31
32 def normalize_melspec(melspec, norm_range=(0, 1)):
33     scaler = MinMaxScaler(feature_range=norm_range)
34     melspec = melspec.T
35     melspec_normalized = scaler.fit_transform(melspec)
36
37     return melspec_normalized.T
38
39
40 def denormalize_melspec(melspec_normalized, original_melspec, norm_range=(0, 1
41                          )):
42     scaler = MinMaxScaler(feature_range=norm_range)
43     melspec = original_melspec.T
44     scaler.fit(melspec)
45     melspec_denormalized = scaler.inverse_transform(melspec_normalized.T)
46
47     return melspec_denormalized.T
48
49 def melspec_to_audio(melspec, sr, n_fft=2048, hop_length=512, n_iter=64):
50     if np.any(melspec < 0):
51         melspec = librosa.db_to_power(melspec)
52
53     audio_reconstructed = librosa.feature.inverse.mel_to_audio(
54         melspec,
55         sr=sr,
56         n_fft=n_fft,
57         hop_length=hop_length,
58         win_length=None,
59         window="hann",
60         center=True,
61         pad_mode="reflect",
62         power=2.0,
63         n_iter=n_iter
64     )
65     return audio_reconstructed
66
67 def display_audio_files(reconstructed_audio, sr, title="", original_audio=None
68                          ):
69     if original_audio is not None:
70         print("Original Audio:")
71         ipd.display(ipd.Audio(original_audio, rate=sr))
72         print("Reconstructed Audio (from Mel Spectrogram):")
73     else:
74         print(title)

```

```

74     ipd.display(ipd.Audio(reconstructed_audio, rate=sr))
75
76
77 def show_spectrogram(spectrogram, title="Mel-Spectrogram", denormalize=False,
78                       is_numpy=False):
79     if not is_numpy:
80         spectrogram = spectrogram.squeeze().cpu().numpy()
81     plt.figure(figsize=(10, 4))
82     if denormalize:
83         plt.imshow(spectrogram, aspect="auto", origin="lower", cmap="viridis")
84     else:
85         plt.imshow(spectrogram, aspect="auto", origin="lower", cmap="viridis",
86                   vmin=0, vmax=1)
87     plt.title(title)
88     plt.xlabel("Time")
89     plt.ylabel("Mel Frequency")
90     plt.colorbar()
91     plt.show()

```



Hình 10: Biểu diễn phổ tần số Mel của tín hiệu âm thanh theo thời gian, với các màu sắc phản ánh cường độ tần số ở từng điểm thời gian.

Tiếp đến, chúng ta duyệt qua các tệp json chứa thông tin về tệp âm thanh tương ứng, để xem có tổng cộng bao nhiêu thể loại trên toàn bộ tập dữ liệu.

```

1 json_dir = os.path.join("piano_audio_files", "crawled_data")
2 all_genres = []
3
4 for filename in os.listdir(json_dir):
5     if filename.endswith('.json'):
6         json_path = os.path.join(json_dir, filename)
7         genres = load_and_get_genres(json_path)

```

```

8         all_genres.extend(genres)
9
10 unique_genres = set(all_genres)
11 max_genres = len(unique_genres)
12 print(f"Total unique genres: {max_genres}")
13 print(f"Unique genres: {unique_genres}")

```

Ngoài ra, để bổ sung thông tin về các thể loại nhạc được gán nhãn cho file âm thanh, ta sử dụng One Hot Encoding. Do đó, cần định nghĩa thêm một số hàm sau:

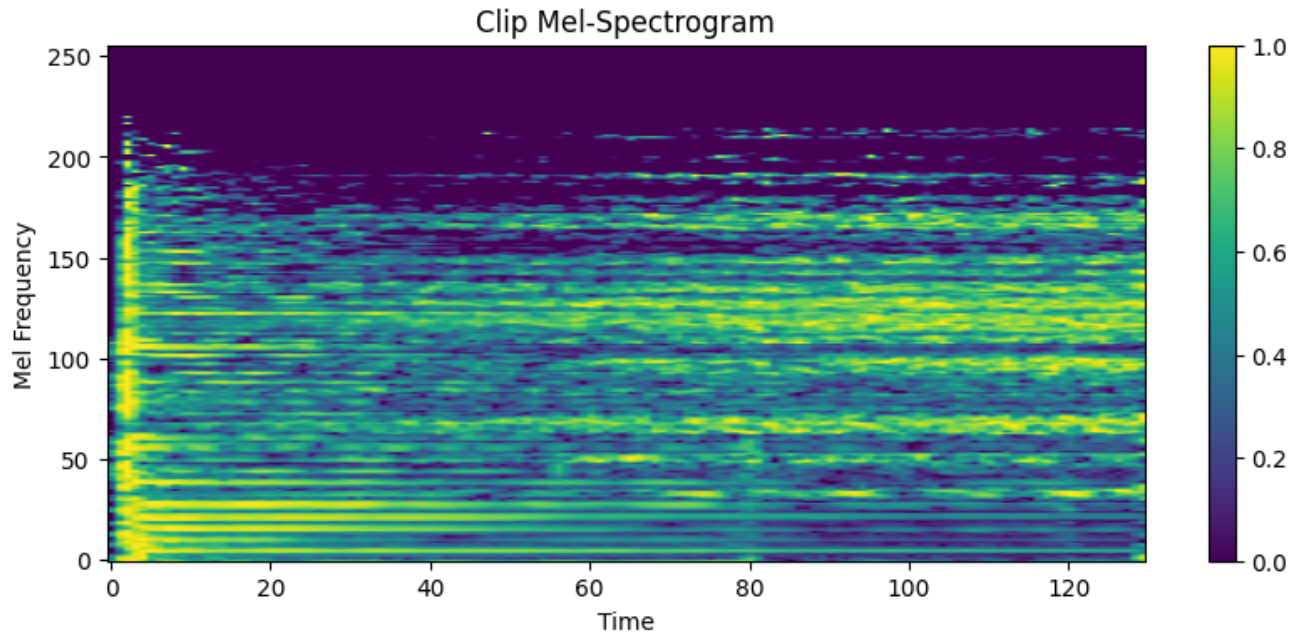
```

1 genres2idx = {genre: idx for idx, genre in enumerate(unique_genres)}
2 idx2genres = {idx: genre for genre, idx in genres2idx.items()}
3
4 def tokenize(genres):
5     return [genres2idx[genre] for genre in genres if genre in genres2idx]
6
7 def detokenize_tolist(tokens):
8     return [idx2genres[token] for token in tokens if token in idx2genres]
9
10 def onehot_encode(tokens, max_genres):
11     onehot = np.zeros(max_genres)
12     onehot[tokens] = 1
13     return onehot
14
15 def onehot_decode(onehot):
16     return [idx for idx, val in enumerate(onehot) if val == 1]

```

II.2.4. Xây dựng PyTorch dataset

Trong phần này chúng ta sẽ xây dựng một lớp Dataset để đọc các tệp âm thanh từ *data_dir* và thể loại tương ứng từ *json_dir*. Với danh sách thể loại đọc được từ tệp json, ta chuyển về dạng onehot vector, trong khi mỗi tệp âm thanh sẽ được chia nhỏ thành các đoạn âm thanh ngắn hơn, cụ thể là 3 giây (*duration*). Như vậy, mỗi mẫu dữ liệu được lấy ra từ Dataset này gồm: đoạn âm thanh 3 giây đã chuẩn hoá *mel_spec_norm*; thể loại *genres_input* và đoạn âm thanh 3 giây chưa chuẩn hoá *mel_spec* dùng để giải chuẩn hoá sau này. Ngoài ra, ta sẽ trích một phần dữ liệu cho tập test.



Hình 11: Hình ảnh spectrogram của một mẫu âm thanh 3 giây được chuẩn hoá.

```

1 class AudioDataset(Dataset):
2     def __init__(self, data_dir, json_dir, sample_rate, duration, n_mels,
3                     n_genres, testset_amount=10):
4         self.data_dir = data_dir
5         self.files = [os.path.join(data_dir, f) for f in os.listdir(data_dir)
6                         if f.endswith(".mp3")]
7         self.json_dir = json_dir
8         self.json_files = [os.path.join(json_dir, f) for f in os.listdir(
9                             json_dir) if f.endswith(".json")]
10        self.sample_rate = sample_rate
11        self.duration = duration
12        self.fixed_length = sample_rate * duration
13        self.n_genres = n_genres
14        self.n_mels = n_mels
15
16        audios = []
17        for file_path, json_file_path in tqdm(zip(self.files, self.json_files)
18                                                , desc=f"Loading audio files in {
19                                                        data_dir}", unit="file", total=len(self
20                                                        .files)):
21            audio, sr = load_and_resample_audio(file_path, target_sr=
22                                                    sample_rate)
23            genres_list = load_and_get_genres(json_file_path)
24
25            genres_tokens = tokenize(genres_list)
26            genres_input = onehot_encode(genres_tokens, n_genres)
27            genres_input = torch.tensor(genres_input, dtype=torch.long).
28                               unsqueeze(0)
29
30            n_samples = len(audio)
31            n_segments = n_samples // self.fixed_length

```

```

24
25         for i in range(n_segments):
26             start = i * self.fixed_length
27             end = (i + 1) * self.fixed_length
28             segment = audio[start:end]
29             mel_spec = audio_to_melspec(segment, sr, self.n_mels, to_db=
30                                     True)
31             mel_spec_norm = normalize_melspec(mel_spec)
32             mel_spec = torch.tensor(mel_spec, dtype=torch.float32).
33                             unsqueeze(0)
34             mel_spec_norm = torch.tensor(mel_spec_norm, dtype=torch.
35                             float32).unsqueeze(0)
36             audios.append((mel_spec_norm, genres_input, mel_spec))
37
38         self.audios = audios[:len(audios) - testset_amount]
39         self.testset = audios[len(audios) - testset_amount:]
40         print(f"Loaded {len(self.audios)} audio segments from {len(self.files)}
41             files, each with shape: {self.audios[0][0].shape}, {self.audios[0][1].shape}
42             , duration: {duration} seconds")
43         print(f"Test set: {len(self.testset)} audio segments")
44
45     def __len__(self):
46         return len(self.audios)
47
48     def __getitem__(self, idx):
49         mel_spec_part, genres_input, mel_spec = self.audios[idx]
50         return mel_spec_part, genres_input, mel_spec

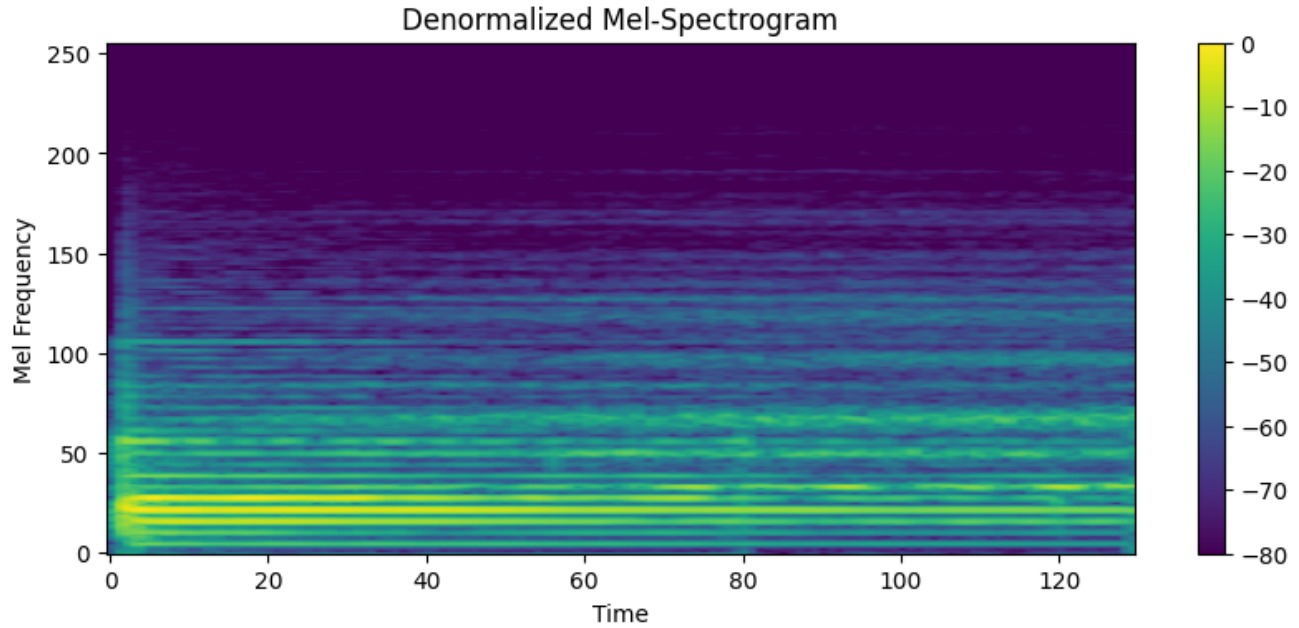
```

Đến đây, ta khởi tạo lớp AudioDataset như đã giải thích bên trên. Sau đó, tạo trainloader và testloader.

```

1 sample_rate = 22050
2 duration = 3
3 n_mels = 256
4 batch_size = 128
5
6 audio_dir = os.path.join("piano_audio_files", "crawled_data", "audio")
7 json_dir = os.path.join("piano_audio_files", "crawled_data")
8
9 testset_amount = 32
10 trainset = AudioDataset(audio_dir, json_dir, sample_rate, duration,
11                         n_mels, max_genres, testset_amount=testset_amount)
12 testset = trainset.testset
13
14 if len(trainset) == 0:
15     raise ValueError(f"No .wav file found in {audio_dir}.")
16
17 trainloader = DataLoader(trainset, batch_size=batch_size, shuffle=True,
18                          num_workers=2)
19 testloader = DataLoader(testset, batch_size=testset_amount, shuffle=False,
20                          num_workers=2)

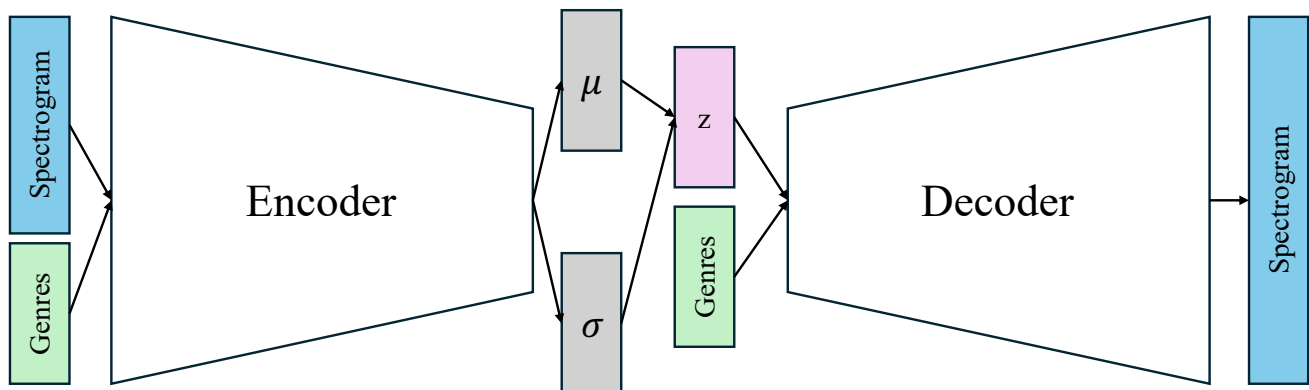
```



Hình 12: Hình ảnh spectrogram sau khi được khử chuẩn hoá từ hình spectrogram của hình 11.

II.2.5. Xây dựng mô hình Conditional Variational Autoencoder (CVAE)

Trong project này, chúng ta sẽ dùng Conditional Variational Autoencoder (CVAE) nhằm mục đích tái tạo lại mẫu âm thanh nhưng với một thể loại khác. Tổng quan, kiến trúc mô hình sẽ gồm Encoder với 3 lớp Conv2d, đầu ra của mỗi lớp Conv2d có kích thước giảm gấp 2 lần; Latent space gồm 2 lớp Linear cho μ (giá trị trung bình) và $\log\sigma$ (log của phương sai); Decoder sẽ tương tự như Encoder nhưng có số chiều và kích thước ngược lại. Ngoài ra, ta dùng thêm Linear cho Decode (`decoder_input`) để chuyển về shape phù hợp với đầu vào. Cuối cùng, hàm Sigmoid để đầu ra có giá trị từ 0 tới 1 (giống các giá trị của đầu vào đã chuẩn hoá).



Hình 13: Minh họa kiến trúc CVAE với dữ liệu đầu vào là spectrogram và genres.

```

1 class CVAE(nn.Module):
2     def __init__(self, d_model, latent_dim, n_frames, n_mels, n_genres):
3         super(CVAE, self).__init__()
4         self.d_model = d_model

```



```

5     self.latent_dim = latent_dim
6     self.n_frames = int(np.ceil(n_frames / 2**3))
7     self.n_mels = int(np.ceil(n_mels / 2**3))
8     self.n_genres = n_genres
9     print(self.n_frames, self.n_mels)
10
11     # Encoder
12     self.encoder = nn.Sequential(
13         nn.Conv2d(1 + self.n_genres, d_model, kernel_size=3, stride=2,
14                     padding=1),
15         nn.BatchNorm2d(d_model),
16         nn.SiLU(),
17         nn.Dropout2d(0.05),
18
19         nn.Conv2d(d_model, d_model * 2, kernel_size=3, stride=2, padding=1),
20         nn.BatchNorm2d(d_model * 2),
21         nn.SiLU(),
22         nn.Dropout2d(0.1),
23
24         nn.Conv2d(d_model * 2, d_model * 4, kernel_size=3, stride=2,
25                     padding=1),
26         nn.BatchNorm2d(d_model * 4),
27         nn.SiLU(),
28         nn.Dropout2d(0.15),
29
30         nn.AdaptiveAvgPool2d((1, 1)), # [B, 4*d, 1, 1]
31         nn.Flatten()
32     )
33
34     # Latent space
35     self.fc_mu = nn.Linear(d_model * 4, latent_dim)
36     self.fc_logvar = nn.Linear(d_model * 4, latent_dim)
37
38     # Decoder
39     self.decoder_input = nn.Linear(latent_dim + self.n_genres, d_model * 4
40                                     * self.n_frames * self.n_mels)
41
42     self.decoder = nn.Sequential(
43         nn.ConvTranspose2d(d_model * 4, d_model * 2, kernel_size=3, stride
44                             =2, padding=1, output_padding=(1, 0)),
45         nn.BatchNorm2d(d_model * 2),
46         nn.SiLU(),
47         nn.Dropout2d(0.1),
48
49         nn.ConvTranspose2d(d_model * 2, d_model, kernel_size=3, stride=2,
50                             padding=1, output_padding=(1, 0)),
51         nn.BatchNorm2d(d_model),
52         nn.SiLU(),
53         nn.Dropout2d(0.05),
54
55         nn.ConvTranspose2d(d_model, 1, kernel_size=3, stride=2, padding=1,
56                             output_padding=1),
57         nn.Sigmoid()
58     )

```

```

53     def reparameterize(self, mu, logvar):
54         std = torch.exp(0.5 * logvar)
55         eps = torch.randn_like(std)
56         return mu + eps * std
57
58     def forward(self, x, genres_input):
59         ori_genres_embed = genres_input.view(genres_input.size(0), -1)
60         genres_embed = ori_genres_embed.unsqueeze(-1).unsqueeze(-1)
61         genres_embed = genres_embed.expand(-1, -1, x.size(2), x.size(3))
62         x_genres = torch.cat((x, genres_embed), dim=1)
63
64         h = x_genres
65         shortcuts = []
66         for block in self.encoder:
67             h = block(h)
68             if isinstance(block, nn.SiLU):
69                 shortcuts.append(h)
70
71         mu = self.fc_mu(h)
72         logvar = self.fc_logvar(h)
73
74         z = self.reparameterize(mu, logvar)
75         z_genres = torch.cat((z, ori_genres_embed), dim=1)
76
77         h_dec = self.decoder_input(z_genres)
78         h_dec = h_dec.view(-1, self.d_model * 4, self.n_frames, self.n_mels)
79
80         for block in self.decoder:
81             if isinstance(block, nn.ConvTranspose2d) and shortcuts:
82                 shortcut = shortcuts.pop()
83                 h_dec = h_dec + shortcut
84             h_dec = block(h_dec)
85
86         recon = h_dec[:, :, :x.size(2), :x.size(3)]
87         return recon, mu, logvar

```

II.2.6. Huấn luyện mô hình

Vì PyTorch không hỗ trợ trực tiếp hàm loss có sẵn cho VAE, chúng ta cần phải tự định nghĩa hàm loss này, đó là sự kết hợp giữa Reconstruction loss và KL-Divergence loss. Code triển khai như sau:

```

1 def loss_function(recon_x, x, mu, logvar):
2     recon_loss = nn.functional.mse_loss(recon_x, x, reduction="sum")
3     KLD = -0.5 * torch.sum(1 + logvar - mu.pow(2) - logvar.exp())
4     return recon_loss + KLD

```

Khi đã có hàm loss, ta tiến hành xây dựng một hàm huấn luyện mô hình cơ bản như sau:

```

1 def train_vae(model, dataloader, optimizer, scheduler, num_epochs,
2               verbose_interval=50):
3     model.train()
4     losses = []
5     for epoch in tqdm(range(num_epochs), desc="Training", unit="epoch"):
6         train_loss = 0

```

```

6         for batch_idx, (data, genres_input, ori_data) in enumerate(dataloader)
7             :
8                 data = data.to(device)
9                 genres_input = genres_input.to(device)
10
11                 optimizer.zero_grad()
12
13                 recon, mu, logvar = model(data, genres_input)
14                 loss = loss_function(recon, data, mu, logvar)
15                 loss.backward()
16                 train_loss += loss.item()
17                 optimizer.step()
18
19                 scheduler.step()
20                 avg_loss = train_loss / len(dataloader.dataset)
21                 losses.append(avg_loss)
22                 print(f"Epoch {epoch}/{num_epochs}, Loss: {avg_loss:.4f}, Lr: {
23                     scheduler.get_last_lr()[0]}")
24                 if epoch == 0 or (epoch + 1) % verbose_interval == 0:
25                     data = data[0].detach().cpu()
26                     recon_img = recon[0].detach().cpu()
27                     show_spectrogram(data, title="Original Spectrogram")
28                     show_spectrogram(recon_img, title="Reconstructed Spectrogram")
29
30         return mu, logvar, losses

```

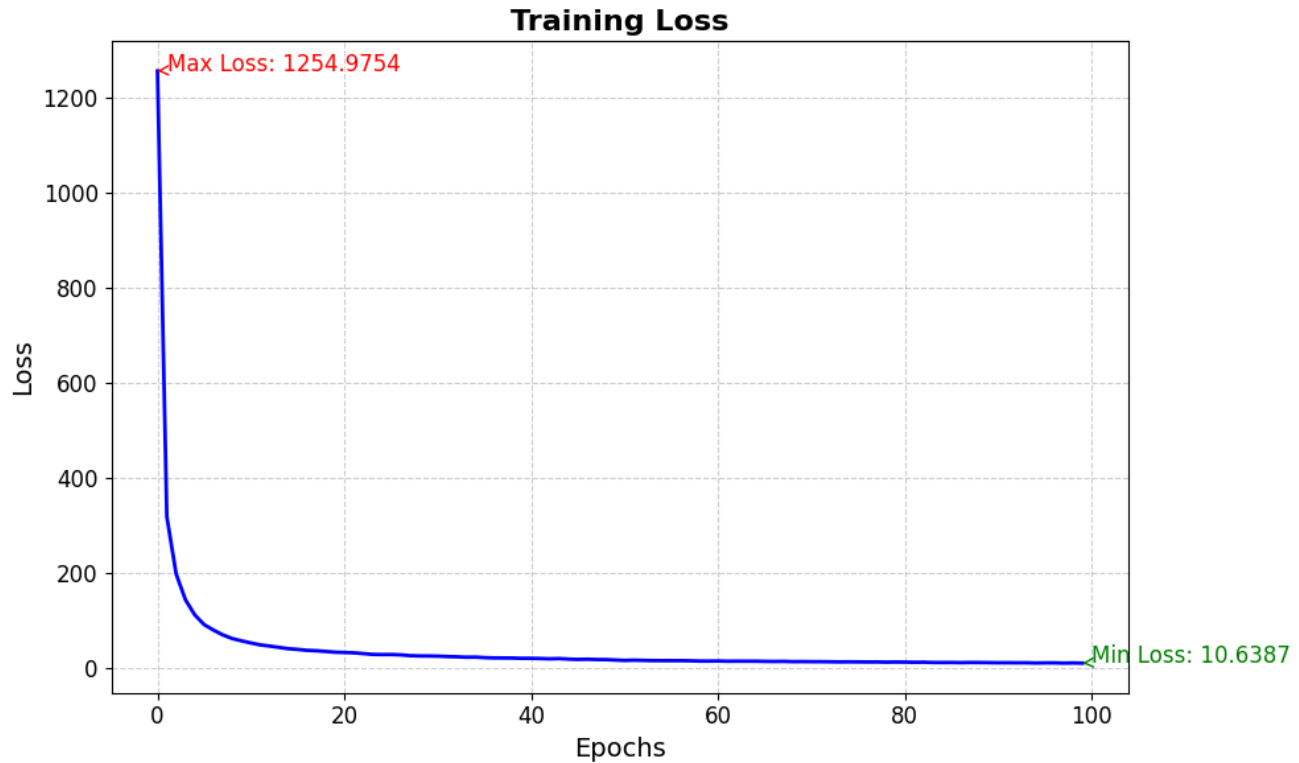
Cuối cùng, ta khai báo giá trị cho một vài tham số liên quan đến mô hình và việc huấn luyện để tiến hành thực hiện lời gọi hàm:

```

1 d_model = 64
2 latent_dim = 128
3 lr = 2e-4
4 num_epochs = 100
5 step_size = num_epochs//2
6 verbose_interval = num_epochs//10
7 gamma = 0.5
8
9 model = CVAE(d_model, latent_dim, n_mels, frame, max_genres).to(device)
10 optimizer = optim.AdamW(model.parameters(), lr=lr)
11 scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=step_size, gamma=
12     gamma)
13 print(f"Total number of parameters: {sum(p.numel() for p in model.parameters()
14     )}")
15 mu, logvar, losses = train_vae(model, trainloader, optimizer, scheduler,
16     num_epochs, verbose_interval=
17     verbose_interval)

```

Khi quá trình huấn luyện kết thúc, ta có được một mô hình với kết quả huấn luyện được trực quan hóa giá trị loss như sau:



Hình 14: Trực quan hóa giá trị loss xuyên suốt quá trình huấn luyện mô hình.

II.2.7. Inference

Để sử dụng mô hình đã huấn luyện tạo ra dữ liệu âm thanh mới, ta triển khai hàm thực hiện inference với đầu vào là một mẫu dữ liệu âm thanh và một danh sách các thể loại nhạc mà ta muốn chuyển đổi dữ liệu gốc thành. Trong trường hợp này, ta triển khai hàm với dữ liệu lấy từ testloader như sau:

```

1 def generate(model, dataloader, genres_list, num_samples=5, diff_level=1):
2     model.eval()
3     with torch.no_grad():
4         data, old_genres_input, ori_data = next(iter(dataloader))
5         data = data.to(device)
6
7         genres_tokens = tokenize(genres_list)
8         genres_input = onehot_encode(genres_tokens, model.n_genres)
9         genres_input = torch.tensor(genres_input, dtype=torch.long).unsqueeze(
10             0)
11         genres_input = genres_input.repeat(old_genres_input.shape[0], 1)
12         genres_input = genres_input.to(device)
13
14         recon, mu, logvar = model(data, genres_input)
15         ori_audios = []
16         recon_audios = []
17         for i in range(num_samples):
18             old_genres_list = detokenize_tolist(onehot_decode(old_genres_input
19                 [i].squeeze().tolist()))
20             show_spectrogram(data[i], title="Original Spectrogram with Genres:

```

```

19         " + ", ".join(old_genres_list))
20     show_spectrogram(recon[i], title="Reconstructed Spectrogram with
21         Genres: " + ", ".join(genres_list))
22
23     diff_spectrogram = torch.abs(data[i] - recon[i]) * diff_level
24     show_spectrogram(diff_spectrogram, title=f"Difference Spectrogram
25         (|Original - Reconstructed|) * {
26             diff_level}")
27     print("Loss: ", loss_function(recon[i], data[i], mu, logvar).item
28         ())
29
30     spec_denorm = denormalize_melspec(recon[i].cpu().numpy().squeeze()
31         , ori_data[i].cpu().numpy().squeeze())
32     audio_reconstructed = melspec_to_audio(spec_denorm, sr=sample_rate
33         )
34     ori_audio = melspec_to_audio(ori_data[i].cpu().numpy().squeeze(),
35         sr=sample_rate)
36
37     recon_audios.append(audio_reconstructed)
38     ori_audios.append(ori_audio)
39
40     display_audio_files(ori_audio, sample_rate, title="Reconstructed
41         Audio with Genres: " + ", ".join(
42             old_genres_list))
43     display_audio_files(audio_reconstructed, sample_rate, title="
44         Reconstructed Audio with Genres: " + ",
45         ".join(genres_list))
46
47     if num_samples > 1:
48         print("-"*100, "Connect all audio", "-"*100)
49         recon_ori_audios = np.concatenate(ori_audios)
50         display_audio_files(recon_ori_audios, sample_rate, title="Connect
51             all original audio")
52         recon_audios = np.concatenate(recon_audios)
53         display_audio_files(recon_audios, sample_rate, title="Connect all
54             reconstructed audio")

```


II.2.8. Triển khai streamlit

Với mô hình đã huấn luyện được, chúng ta có thể triển khai thành ứng dụng web demo để sử dụng. Hình ảnh dưới đây là một ví dụ xây dựng thông qua streamlit, các bạn có thể truy cập trải nghiệm tại mục [IV](#).



New Genres Audio Reconstruction

Tải lên 1 audio (chỉ xử lý 15s đầu tiên)

 Drag and drop file here
Limit 200MB per file • WAV, MP3

Browse files

Hoặc chọn 1 audio mẫu dưới đây:

classical.00000.wav

▶ 0:00 / 0:30

🔊 ⋮

Chọn thể loại

Soundtrack × Jazz × Christmas × New Age × Chill-out ×

Xử lý Âm Thanh

Kết quả:

▶ 0:00 / 0:14

🔊 ⋮

Hình 15: Minh họa web demo cho chương trình tạo sinh âm thanh dựa trên âm thanh đầu vào và các thể loại nhạc.

III. Câu hỏi trắc nghiệm

1. Phát biểu nào sau đây là đúng về Variational Autoencoder (VAE)?
 - (a) VAE là một mô hình phân loại.
 - (b) VAE là một mô hình sinh dữ liệu.
 - (c) VAE là một mô hình dự đoán một giá trị liên tục.
 - (d) VAE là một mô hình clustering.
2. Reparameterization trick trong VAE dùng để:
 - (a) Tách riêng encoder và decoder.
 - (b) Cho phép gradient lan truyền qua quá trình sampling.
 - (c) Tăng nhiễu cho đa dạng đầu ra.
 - (d) Chuyển dữ liệu liên tục thành rời rạc.
3. Hàm loss của VAE truyền thống bao gồm:
 - (a) Chỉ MSE.
 - (b) Chỉ KL Divergence.
 - (c) Cả MSE và KL Divergence.
 - (d) MSE và Cross Entropy.
4. MSE trong loss của VAE đo lường:
 - (a) Độ lệch trung bình dữ liệu.
 - (b) Lỗi tái tạo (reconstruction error).
 - (c) Sai số dự đoán.
 - (d) Độ nhiễu trong tín hiệu.
5. KL Divergence trong loss của VAE đo lường:
 - (a) Khoảng cách giữa latent distribution và normal distribution.
 - (b) Sai số tái tạo.
 - (c) Độ phân tán dữ liệu.
 - (d) Sự chênh lệch đầu vào - đầu ra.
6. Encoder trong VAE có chức năng:
 - (a) Sinh dữ liệu mới từ latent space.
 - (b) Nén đầu vào thành latent space.
 - (c) Tăng nhiễu dữ liệu.
 - (d) Tạo đầu ra từ latent.
7. Decoder trong VAE có nhiệm vụ:

- (a) Lấy mẫu từ latent và tái tạo đầu ra.
- (b) Nén dữ liệu vào latent.
- (c) Tính toán hàm loss.
- (d) Huấn luyện mô hình.

8. Thư viện **librosa** được dùng để:

```
1 import librosa
2 audio, sr = librosa.load('example.mp3', sr=22050)
3 mel_spec = librosa.feature.melspectrogram(y=audio, sr=sr)
```

- (a) Chuyển audio thành văn bản.
- (b) Tính toán đặc trưng âm thanh.
- (c) Phục hồi ảnh từ audio.
- (d) Trực quan hóa tín hiệu.

9. Hàm `librosa.feature.melspectrogram` dùng để:

```
1 mel_spec = librosa.feature.melspectrogram(y=audio, sr=sr, n_mels=128)
```

- (a) Tính toán mel-spectrogram từ audio.
- (b) Chuyển đổi mel-spectrogram thành audio.
- (c) Hiển thị biểu đồ năng lượng.
- (d) Chuẩn hóa dữ liệu.

10. Tại sao khi code lại dùng `logvar` thay vì dùng giá trị variance trực tiếp?

```
1 KLD = -0.5 * torch.sum(1 + logvar - mu.pow(2) - logvar.exp())
```

- (a) Giúp ổn định số học, gradient hiệu quả.
- (b) Giảm tham số mô hình, tăng tốc huấn luyện.
- (c) Gây khó khăn trong quá trình lan truyền ngược (backpropagation), dễ tràn số.
- (d) Tất cả các đáp án trên.

IV. Phụ lục

1. **Datasets:** Các file dataset được đề cập trong bài có thể được tải tại [đây](#).
2. **Hint:** Các file code gợi ý có thể được tải tại [đây](#).
3. **Solution:** Các file code cài đặt hoàn chỉnh và phần trả lời nội dung trắc nghiệm có thể được tải tại [đây](#) (**Lưu ý:** Sáng thứ 3 khi hết deadline phần project, ad mới copy các nội dung bài giải nêu trên vào đường dẫn).
4. **Demo:** Web demo và mã nguồn của ứng dụng có thể được truy cập tại [đây](#).
5. **Rubric:**

Mục	Kiến Thức	Đánh Giá
I.	<ul style="list-style-type: none"> - Kiến thức về thu thập dữ liệu thông qua kỹ thuật crawling data. - Kiến thức về thư viện Selenium. - Kiến thức cơ bản về HTML và cách xây dựng logic thu thập dữ liệu thông qua cấu trúc HTML của một trang web. 	<ul style="list-style-type: none"> - Nắm được cách sử dụng thư viện Selenium để lấy dữ liệu từ một trang web. - Có khả năng phân tích cơ bản cấu trúc HTML để xây dựng từng bước xử lý để bóc tách được dữ liệu cần thu thập từ trang web.
II.	<ul style="list-style-type: none"> - Kiến trúc Variational Autoencoder (VAE) và Conditional VAE (CVAE). - Các kỹ thuật xử lý tín hiệu âm thanh cơ bản. - Cách xây dựng mô hình VAE sử dụng thư viện PyTorch. - Cách chuẩn bị và tổ chức dữ liệu huấn luyện âm thanh và văn bản cho VAE để có khả năng tái tạo âm nhạc. 	<ul style="list-style-type: none"> - Nắm được các lý thuyết cơ bản về kiến trúc VAE và CVAE. - Khả năng lập trình PyTorch để xây dựng mô hình VAE và CVAE. - Cách ứng dụng CVAE trên dữ liệu âm thanh và văn bản để triển khai mô hình tái tạo ảnh spectrogram của âm thanh.

- Hết -