Tải dữ liệu âm thanh

Bạn có thể tải tập dữ liệu âm thanh bằng tính năng Âm thanh tự động giải mã và lấy mẫu lại các tệp âm thanh khi bạn truy cập vào các ví dụ.
Giải mã âm thanh dựa trên gói python torchcodec, sử dụng thư viện FFmpeg C dưới mui xe.

Cài đặt

Để làm việc với tập dữ liệu âm thanh, bạn cần cài đặt phần phụ thuộc âm thanh.
Hãy xem hướng dẫn cài đặt để tìm hiểu cách cài đặt nó.

Tệp cục bộ

You can load your own dataset using the paths to your audio files. Use the cast_column() để lấy một cột đường dẫn tệp âm thanh và chuyển nó sang tính năng Âm thanh:

```
>>> audio_dataset = Dataset.from_dict({"audio": ["path/to/audio_1", "path/to/audio_2", ..., "path/
>>> audio_dataset[0]["audio"]
<datasets.features._torchcodec.AudioDecoder object at 0x11642b6a0>
```

Thư mục âm thanh

Bạn cũng có thể tải tập dữ liệu bằng trình tạo tập dữ liệu AudioFolder. Nó không yêu cầu viết một trình tải dữ liệu tùy chỉnh, giúp nó hữu ích trong việc tạo và tải nhanh các tập dữ liệu âm thanh với vài nghìn tập tin âm thanh.

AudioFolder với siêu dữ liệu

Để liên kết các tệp âm thanh của bạn với thông tin siêu dữ liệu, hãy đảm bảo tập dữ liệu của bạn có siêu dữ l tài liệu. Cấu trúc tập dữ liệu của bạn có thể trông giống như:

```
thư mục/train/metadata.csv
thư mục/train/first_audio_file.mp3
thư mục/train/second_audio_file.mp3
thư mục/train/third_audio_file.mp3
```

Tệp siêu dữ liệu.csv của bạn phải có cột file_name liên kết các tệp âm thanh với chúng
siêu dữ liệu. Một tệp siêu dữ liệu.csv mẫu có thể trông giống như:

```
file_name,bản phiên âm
first_audio_file.mp3, tinh thần và thể xác sẽ lại hợp nhất trong mùa xuân trẻ trung, bạn sẽ trỗi dậy và bạn c
giây_audio_file.mp3, đã ở cổng źwierzyniec, nhà vua ngồi cùng với ông ấy, hội đồng hoàng tử và lãnh chúa
Third_audio_file.mp3, chắc chúng ta điên cuồng đâu đó rồi, đợi ngày gửi thôi, sz
```

AudioFolder sẽ tải dữ liệu âm thanh và tạo cột phiên âm chứa văn bản từ
siêu dữ liệu.csv :

```
>>> from datasets import load_dataset

>>> dataset = load_dataset("username/dataset_name")
>>> # OR locally:
>>> dataset = load_dataset("/path/to/folder")
```

For local datasets, this is equivalent to passing audiofolder manually in load_dataset() and
thư mục trong data_dir :

```
>>> dataset = load_dataset("audiofolder", data_dir="/path/to/folder")
```

Siêu dữ liệu cũng có thể được chỉ định dưới dạng Dòng JSON, trong trường hợp đó, hãy sử dụng siêu dữ liệu
tên của tệp siêu dữ liệu. Định dạng này hữu ích trong các trường hợp khi một trong các cột bị
phức tạp, ví dụ một danh sách các số float, để tránh lỗi phân tích cú pháp hoặc đọc các giá trị phức tạp dưới

To ignore the information in the metadata file, set drop_metadata=True in load_dataset():

```
>>> from datasets import load_dataset

>>> dataset = load_dataset("username/dataset_with_metadata", drop_metadata=True)
```

Nếu bạn không có tệp siêu dữ liệu, AudioFolder sẽ tự động suy ra tên nhãn từ
tên thư mục.
If you want to drop automatically created labels, set drop_labels=True .
Trong trường hợp này, tập dữ liệu của bạn sẽ chỉ chứa một cột âm thanh:

```
>>> from datasets import load_dataset

>>> dataset = load_dataset("username/dataset_without_metadata", drop_labels=True)
```

Cuối cùng, đối số bộ lọc cho phép bạn chỉ tải một tập hợp con của tập dữ liệu, dựa trên một điều kiện
trên nhãn hoặc siêu dữ liệu. Điều này đặc biệt hữu ích nếu siêu dữ liệu ở định dạng Parquet,
vì định dạng này cho phép lọc nhanh. Bạn cũng nên sử dụng đối số này với
streaming=True , because by default the dataset is fully downloaded before filtering.

```
>>> filters = [("label", "=", 0)]
>>> dataset = load_dataset("username/dataset_name", streaming=True, filters=filters)
```

> [!TIP]
> Để biết thêm thông tin về cách tạo tập dữ liệu AudioFolder của riêng bạn, hãy xem
> Tạo hướng dẫn tập dữ liệu âm thanh.

Để biết hướng dẫn về cách tải bất kỳ loại tập dữ liệu nào, hãy xem hướng dẫn tải chung.

Giải mã âm thanh

Theo mặc định, các tập tin âm thanh được giải mã tuần tự dưới dạng đối tượng torchcodec AudioDecoding k
lặp lại trên một tập dữ liệu.
Tuy nhiên, có thể tăng tốc đáng kể tập dữ liệu bằng cách sử dụng giải mã đa luồng:

```
>>> import os
>>> num_threads = num_threads = min(32, (os.cpu_count() or 1) + 4)
>>> dataset = dataset.decode(num_threads=num_threads)
>>> for example in dataset:# up to 20 times faster !
None
```

Bạn có thể kích hoạt đa luồng bằng cách sử dụng num_threads. Điều này đặc biệt hữu ích để tăng tốc

truyền dữ liệu từ xa.
However it can be slower than num_threads=0 for local data on fast disks.

Nếu bạn không quan tâm đến các hình ảnh được giải mã dưới dạng mảng NumPy và muốn truy cập thay vào đó, path/byte, bạn có thể tắt giải mã:

```
>>> dataset = dataset.decode(False)
```

Note: IterableDataset.decode() is only available for streaming datasets at the moment.