Tải dữ liệu video

[!WARNING]
Hỗ trợ video là thử nghiệm và có thể thay đổi.

Bộ dữ liệu video có các cột loại Video chứa các đối tượng torchvision.

[!TIP]
Để làm việc với bộ dữ liệu video, bạn cần có gói torchvision và av
đã cài đặt. Hãy xem hướng dẫn cài đặt để tìm hiểu cách cài đặt chúng.

Khi bạn tải tập dữ liệu video và gọi cột video, các video sẽ được giải mã thành
video ngọn đuốc:

```
>>> from datasets import load_dataset, Video

>>> dataset = load_dataset("path/to/video/folder", split="train")
>>> dataset[0]["video"]
<torchcodec.decoders._video_decoder.VideoDecoder object at 0x14a61d5a0>
```

[!WARNING]
Lập chỉ mục vào tập dữ liệu video bằng cách sử dụng chỉ mục hàng trước rồi đến cột video - dataset[0]["video"] - to avoid creating all the video objects in the dataset. Otherwise,
đây có thể là một quá trình chậm và tốn thời gian nếu bạn có một tập dữ liệu lớn.

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

Đọc khung

Access frames directly from a video using the VideoReader using next() :

```
>>> video = dataset[0]["video"]
>>> first_frame = video.get_frame_at(0)
>>> first_frame.data.shape
(3, 240, 320)
>>> first_frame.pts_seconds# timestamp
0,0
```

Để có được nhiều khung hình cùng một lúc, bạn có thể gọi
.get_frames_in_range(start: int, stop: int, step: int) . This will return a frame batch.
Đây là cách hiệu quả để có được danh sách khung dài, hãy tham khảo tài liệu torchcodec để xem thêm chức năng truy cập dữ liệu hiệu quả:

```
>>> import torch
>>> frames = video.get_frames_in_range(0, 6, 1)
>>> frames.data.shape
torch.Size([5, 3, 240, 320])
```

There is also .get_frames_played_in_range(start_seconds: float, stop_seconds: float) to
truy cập tất cả các khung hình được phát trong một khoảng thời gian nhất định.

```
>>> frames = video.get_frames_played_in_range(.5, 1.2)
>>> frames.data.shape
torch.Size([42, 3, 240, 320])
```

Tệp cục bộ

You can load a dataset from the video path. Use the cast_column() function to accept a column
đường dẫn tệp video và giải mã nó thành video torchcodec với tính năng Video:

```
>>> from datasets import Dataset, Video

>>> dataset = Dataset.from_dict({"video": ["path/to/video_1", "path/to/video_2", ..., "path/to/vid
>>> dataset[0]["video"]
<torchcodec.decoders._video_decoder.VideoDecoder object at 0x14a61e080>
```

Nếu bạn chỉ muốn tải đường dẫn cơ bản tới tập dữ liệu video mà không giải mã video object, set decode=False in the Video feature:

```
>>> dataset = dataset.cast_column("video", Video(decode=False))
>>> dataset[0]["video"]
{'bytes': None,
 'path': 'path/to/video/folder/video0.mp4'}
```

Thư mục Video

Bạn cũng có thể tải tập dữ liệu bằng trình tạo tập dữ liệu VideoFolder mà không yêu cầu viết một trình tải dữ liệu tùy chỉnh. Điều này làm cho VideoFolder trở nên lý tưởng để tạo và tải nhanh chóng bộ dữ liệu video với hàng nghìn video cho các nhiệm vụ thị giác khác nhau. Tập dữ liệu video của bạn cấu trúc sẽ trông như thế này:

```
thư mục/xe lửa/chó/golden_retriever.mp4
folder/train/dog/german_shepherd.mp4
folder/train/dog/chihuahua.mp4

thư mục/train/cat/maine_coon.mp4
thư mục/train/cat/bengal.mp4
folder/train/cat/birman.mp4
```

Nếu tập dữ liệu tuân theo cấu trúc VideoFolder thì bạn có thể tải nó trực tiếp bằng load_dataset():

```
>>> 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 videofolder manually in load_dataset() and thư mục trong data_dir :

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

Sau đó, bạn có thể truy cập các video dưới dạng torchcodec.decodings._video_decoding.VideoDecoding đối tượng:

```
>>> dataset["train"][0]
{"video": <torchcodec.decoders._video_decoder.VideoDecoder object at 0x14a61e080>, "label": 0}

>>> dataset["train"][-1]
{"video": <torchcodec.decoders._video_decoder.VideoDecoder object at 0x14a61e090>, "label": 1}
```

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, VideoFolder 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 video:

```
>>> 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 VideoFolder của riêng bạn, hãy xem Tạo hướng dẫn tập dữ liệu video.

Bộ dữ liệu Web

Định dạng WebDataset dựa trên một thư mục lưu trữ TAR và phù hợp với video lớn bộ dữ liệu.
Because of their size, WebDatasets are generally loaded in streaming mode (using streaming=True).

Bạn có thể tải WebDataset như thế này:

```
>>> from datasets import load_dataset

>>> dataset = load_dataset("webdataset", data_dir="/path/to/folder", streaming=True)
```

Giải mã video

Theo mặc định, video được giải mã tuần tự dưới dạng Trình đọc video torchvision khi bạn lặp lại một tập dữ liệu.
Nó giải mã tuần tự siêu dữ liệu của video và không đọc các khung hình video cho đến khi bạn truy cập chúng.

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 video được giải mã dưới dạng VideoReaders torchvision và muốn

thay vào đó, hãy truy cập vào đường dẫn/byte, bạn có thể tắt giải mã:

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

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