Sử dụng với JAX

Tài liệu này là phần giới thiệu nhanh về cách sử dụng bộ dữ liệu với JAX, đặc biệt tập trung vào
làm thế nào để có được
các đối tượng jax.Array ngoài bộ dữ liệu của chúng tôi và cách sử dụng chúng để huấn luyện các mô hình JAX

[!TIP]
jax và jaxlib được yêu cầu sao chép lại mã ở trên, vì vậy hãy đảm bảo rằng bạn
install them as pip install datasets[jax] .

Định dạng tập dữ liệu

Theo mặc định, bộ dữ liệu trả về các đối tượng Python thông thường: số nguyên, số float, chuỗi, danh sách, v
các đối tượng chuỗi và nhị phân không thay đổi vì JAX chỉ hỗ trợ số.

To get JAX arrays (numpy-like) instead, you can set the format of the dataset to jax :

```
>>> from datasets import Dataset
>>> data = [[1, 2], [3, 4]]
>>> ds = Dataset.from_dict({"data": data})
>>> ds = ds.with_format("jax")
>>> ds[0]
{'data': DeviceArray([1, 2], dtype=int32)}
>>> ds[:2]
{'data': DeviceArray([
       [1, 2],
       [3, 4]], dtype=int32)}
```

[!TIP]
Đối tượng Dataset là một trình bao bọc của bảng Mũi tên, cho phép đọc nhanh từ các mảng trong
tập dữ liệu vào mảng JAX.

Lưu ý rằng quy trình tương tự cũng áp dụng cho các đối tượng DatasetDict, do đó
khi đặt định dạng của DatasetDict thành jax, tất cả các Dataset ở đó
sẽ được định dạng là jax :

```
>>> from datasets import DatasetDict
>>> data = {"train": {"data": [[1, 2], [3, 4]]}, "test": {"data": [[5, 6], [7, 8]]}}
>>> dds = DatasetDict.from_dict(data)
>>> dds = dds.with_format("jax")
>>> dds["train"][:2]
{'data': DeviceArray([
       [1, 2],
       [3, 4]], dtype=int32)}
```

Một điều khác bạn cần cân nhắc là định dạng không được áp dụng
cho đến khi bạn thực sự truy cập dữ liệu. Vì vậy, nếu bạn muốn lấy một mảng JAX ra khỏi tập dữ liệu, trước tiên bạn cần truy cập dữ liệu, nếu không định dạng sẽ giữ nguyên.

Cuối cùng, để tải dữ liệu vào thiết bị bạn chọn, bạn có thể chỉ định đối số thiết bị,
nhưng lưu ý rằng jaxlib.xla_extension.Device không được hỗ trợ vì nó không thể tuần tự hóa được dưa chua không phải là dill , vì vậy thay vào đó bạn sẽ cần sử dụng mã định danh chuỗi của nó:

```
>>> import jax
>>> from datasets import Dataset
>>> data = [[1, 2], [3, 4]]
>>> ds = Dataset.from_dict({"data": data})
>>> device = str(jax.devices()[0])# Not casting to `str` before passing it to `with_format` will
>>> ds = ds.with_format("jax", device=device)
>>> ds[0]
{'data': DeviceArray([1, 2], dtype=int32)}
>>> ds[0]["data"].device()
TFRT_CPU_0
>>> assert ds[0]["data"].device() == jax.devices()[0]
ĐÚNG VẬY
```

Lưu ý rằng nếu đối số thiết bị không được cung cấp cho with_format thì nó sẽ sử dụng giá trị mặc định device which is jax.devices()[0] .

Mảng N chiều

Nếu tập dữ liệu của bạn bao gồm các mảng N chiều, bạn sẽ thấy theo mặc định chúng là
được coi là cùng một tenxơ nếu hình dạng cố định:

```
>>> from datasets import Dataset
>>> data = [[[1, 2],[3, 4]], [[5, 6],[7, 8]]]# fixed shape
>>> ds = Dataset.from_dict({"data": data})
>>> ds = ds.with_format("jax")
>>> ds[0]
{'data': Array([[1, 2],
                [3, 4]], dtype=int32)}
```

```
>>> from datasets import Dataset
>>> data = [[[1, 2],[3]], [[4, 5, 6],[7, 8]]]# varying shape
>>> ds = Dataset.from_dict({"data": data})
>>> ds = ds.with_format("jax")
>>> ds[0]
{'data': [Array([1, 2], dtype=int32), Array([3], dtype=int32)]}
```

Tuy nhiên logic này thường yêu cầu so sánh hình dạng và sao chép dữ liệu chậm.
Để tránh điều này, bạn phải sử dụng rõ ràng loại tính năng Mảng và chỉ định hình dạng của tensor:

```
>>> from datasets import Dataset, Features, Array2D
>>> data = [[[1, 2],[3, 4]],[[5, 6],[7, 8]]]
>>> features = Features({"data": Array2D(shape=(2, 2), dtype='int32')})
>>> ds = Dataset.from_dict({"data": data}, features=features)
>>> ds = ds.with_format("jax")
>>> ds[0]
{'data': Array([[1, 2],
                [3, 4]], dtype=int32)}
>>> ds[:2]
{'data': Array([[[1, 2],
                 [3, 4]],

                [[5, 6],
                 [7, 8]]], dtype=int32)}
```

Các loại tính năng khác

Dữ liệu ClassLabel được chuyển đổi chính xác thành mảng:

```
>>> from datasets import Dataset, Features, ClassLabel
>>> labels = [0, 0, 1]
>>> features = Features({"label": ClassLabel(names=["negative", "positive"])})
>>> ds = Dataset.from_dict({"label": labels}, features=features)
>>> ds = ds.with_format("jax")
>>> ds[:3]
{'label': DeviceArray([0, 0, 1], dtype=int32)}
```

Các đối tượng chuỗi và nhị phân không thay đổi vì JAX chỉ hỗ trợ số.

Các loại tính năng Hình ảnh và Âm thanh cũng được hỗ trợ.

> [!TIP]
> Để sử dụng loại tính năng Hình ảnh, bạn sẽ cần cài đặt thêm tầm nhìn như `pip install datasets[vision]` .

```
>>> from datasets import Dataset, Features, Image
>>> images = ["path/to/image.png"] * 10
>>> features = Features({"image": Image()})
>>> ds = Dataset.from_dict({"image": images}, features=features)
>>> ds = ds.with_format("jax")
>>> ds[0]["image"].shape
(512, 512, 3)
>>> ds[0]
{'image': DeviceArray([[[ 255, 255, 255],
                        [ 255, 255, 255],
                        None
                        [ 255, 255, 255],
                        [ 255, 255, 255]]], dtype=uint8)}
>>> ds[:2]["image"].shape
(2, 512, 512, 3)
>>> ds[:2]
{'image': DeviceArray([[[[ 255, 255, 255],
                        [ 255, 255, 255],
                        None
                        [ 255, 255, 255],
                        [ 255, 255, 255]]]], dtype=uint8)}
```

[!TIP]
Để sử dụng loại tính năng Âm thanh, bạn sẽ cần cài đặt thêm âm thanh như
pip install datasets[audio] .

```
>>> from datasets import Dataset, Features, Audio
>>> audio = ["path/to/audio.wav"] * 10
>>> features = Features({"audio": Audio()})
>>> ds = Dataset.from_dict({"audio": audio}, features=features)
>>> ds = ds.with_format("jax")
>>> ds[0]["audio"]["array"]
DeviceArray([-0.059021, -0.03894043, -0.00735474, ...,0.0133667 ,
                    0.01809692,0.00268555], dtype=float32)
>>> ds[0]["audio"]["sampling_rate"]
DeviceArray(44100, dtype=int32, weak_type=True)
```

Đang tải dữ liệu

JAX không có bất kỳ khả năng tải dữ liệu tích hợp nào, vì vậy bạn sẽ cần sử dụng thư viện như
dưới dạng PyTorch để tải dữ liệu của bạn bằng DataLoader hoặc TensorFlow
sử dụng tf.data.Dataset . Trích dẫn tài liệu JAX về chủ đề này:
"JAX tập trung vào việc chuyển đổi chương trình và NumPy được hỗ trợ bởi bộ tăng tốc, vì vậy chúng tôi khôn
bao gồm tải hoặc trộn dữ liệu trong thư viện JAX. Đã có rất nhiều dữ liệu tuyệt vời
máy xúc
ngoài kia, vì vậy chúng ta hãy sử dụng chúng thay vì phát minh lại bất cứ thứ gì. Chúng tôi sẽ lấy dữ liệu của
máy xúc,
và tạo một miếng đệm nhỏ để làm cho nó hoạt động với mảng NumPy.".

Vì vậy, đó là lý do tại sao định dạng JAX trong bộ dữ liệu lại hữu ích vì nó cho phép bạn sử dụng
bất kỳ mô hình nào từ HuggingFace Hub với JAX mà không phải lo lắng về dữ liệu
đang tải
phần.

Using with_format('jax')

The easiest way to get JAX arrays out of a dataset is to use the with_format('jax') method.
Hãy giả sử

rằng chúng tôi muốn đào tạo một mạng lưới thần kinh trên bộ dữ liệu MNIST có sẵn
tại HuggingFace Hub tại https://huggingface.co/datasets/mnist.

```
>>> from datasets import load_dataset
>>> ds = load_dataset("mnist")
>>> ds = ds.with_format("jax")
>>> ds["train"][0]
{'image': DeviceArray([[0,0,0, ...],
                       [0,0,0, ...],
                       None
                       [0,0,0, ...],
                       [0,0,0, ...]], dtype=uint8),
 'label': DeviceArray(5, dtype=int32)}
```

Sau khi định dạng được đặt, chúng ta có thể cung cấp tập dữ liệu cho mô hình JAX theo đợt bằng cách sử dụ

Dataset.iter()
phương pháp:

```
>>> for epoch in range(epochs):
...for batch in ds["train"].iter(batch_size=32):
...x, y = batch["image"], batch["label"]
None
```