Xử lý dữ liệu hình ảnh

Hướng dẫn này hiển thị các phương pháp cụ thể để xử lý tập dữ liệu hình ảnh. Tìm hiểu cách:

- Use map() with image dataset.
- Apply data augmentations to a dataset with set_transform().

Để biết hướng dẫn về cách xử lý bất kỳ loại tập dữ liệu nào, hãy xem hướng dẫn quy trình chung.

Bản đồ

The map() function can apply transforms over an entire dataset.

Ví dụ: tạo hàm Thay đổi kích thước cơ bản:

```
>>> def transforms(examples):
...examples["pixel_values"] = [image.convert("RGB").resize((100,100)) for image in examples["
...trả lại ví dụ
```

Now use the map() function to resize the entire dataset, and set batched=True to speed up the xử lý bằng cách chấp nhận hàng loạt ví dụ. Phép biến đổi trả về pixel_values dưới dạng Đối tượng PIL.Image có thể lưu trong bộ nhớ cache:

```
>>> dataset = dataset.map(transforms, remove_columns=["image"], batched=True)
>>> dataset[0]
{'label': 6,
  'pixel_values': <PIL.PngImagePlugin.PngImageFile image mode=RGB size=100x100 at 0x7F058237BB10>}
```

Tệp bộ đệm giúp tiết kiệm thời gian vì bạn không phải thực hiện cùng một phép biến đổi hai lần. các map() function is best for operations you only run once per training - like resizing an image - thay vì sử dụng nó cho các hoạt động được thực hiện cho từng kỷ nguyên, như tăng cường dữ liệu.

map() takes up some memory, but you can reduce its memory requirements with the following thông số:

- batch_size xác định số lượng mẫu được xử lý trong một lệnh gọi tới

hàm biến đổi.

- writer_batch_size xác định số lượng mẫu đã xử lý được lưu giữ trong bộ nhớ trước khi chúng được lưu trữ đi.

Cả hai giá trị tham số đều mặc định là 1000, điều này có thể tốn kém nếu bạn lưu trữ hình ảnh.
Lower these values to use less memory when you use map().

Áp dụng các phép biến đổi

Bộ dữ liệu áp dụng tính năng tăng cường dữ liệu từ bất kỳ thư viện hoặc gói nào cho tập dữ liệu của bạn.
Transforms can be applied on-the-fly on batches of data with set_transform(), which consumes
ít dung lượng ổ đĩa hơn.

[!TIP]
Ví dụ sau sử dụng torchvision, nhưng bạn có thể thoải mái sử dụng tính năng tăng cường dữ liệu khác các thư viện như Albumentations, Kornia và imgaug.

Ví dụ: nếu bạn muốn thay đổi ngẫu nhiên thuộc tính màu của hình ảnh:

```
>>> from torchvision.transforms import Compose, ColorJitter, ToTensor

>>> jitter = Compose(
...[
...ColorJitter(brightness=0.25, contrast=0.25, saturation=0.25, hue=0.7),
...ToTensor(),
...]
... )
```

Tạo một hàm để áp dụng biến đổi ColorJitter:

```
>>> def transforms(examples):
...examples["pixel_values"] = [jitter(image.convert("RGB")) for image in examples["image"]]
...trả lại ví dụ
```

Apply the transform with the set_transform() function:

```
>>> dataset.set_transform(transforms)
```