Bắt đầu nhanh

Hướng dẫn bắt đầu nhanh này dành cho các nhà phát triển đã sẵn sàng tìm hiểu sâu về mã và xem ví dụ về cách tích hợp Bộ dữ liệu vào quy trình đào tạo mô hình của họ. Nếu bạn là người mới bắt đầu, chúng tôi khuyên bạn nên bắt đầu với các hướng dẫn của chúng tôi, nơi bạn sẽ được giới thiệu kỹ lưỡng hơn

Mỗi tập dữ liệu là duy nhất và tùy thuộc vào nhiệm vụ, một số tập dữ liệu có thể yêu cầu bổ sung các bước chuẩn bị cho việc đào tạo. Nhưng bạn luôn có thể sử dụng Công cụ bộ dữ liệu để tải và xử lý tập dữ liệu. Cách nhanh nhất và dễ dàng nhất để bắt đầu là tải tập dữ liệu hiện có từ Ôm mặt Hub. Có hàng nghìn bộ dữ liệu để lựa chọn, trải rộng trên nhiều nhiệm vụ.
Chọn loại tập dữ liệu bạn muốn làm việc và bắt đầu!

Âm thanh
Lấy mẫu lại tập dữ liệu âm thanh và chuẩn bị sẵn sàng cho một mô hình để phân loại loại vấn đề ngân hàng một diễn giả đang gọi về.

Tầm nhìn
Áp dụng tính năng tăng cường dữ liệu cho tập dữ liệu hình ảnh và chuẩn bị sẵn sàng cho mô hình chẩn đoán ở cây đậu.

NLP
Mã hóa một tập dữ liệu và chuẩn bị sẵn sàng cho một mô hình để xác định xem một cặp câu có ý nghĩa tương tự.

> [!TIP]
> Hãy xem Chương 5 của khóa học Ôm Mặt để tìm hiểu thêm về những điều quan trọng khác. các chủ đề như tải tập dữ liệu từ xa hoặc cục bộ, các công cụ để dọn dẹp tập dữ liệu và tạo tập dữ liệu của riêng bạn.

Bắt đầu bằng cách cài đặt Bộ dữ liệu:

```
tập dữ liệu cài đặt pip
```

Bộ dữ liệu còn hỗ trợ các định dạng dữ liệu âm thanh và hình ảnh:

- Để làm việc với tập dữ liệu âm thanh, hãy cài đặt tính năng Âm thanh:

pip install datasets[audio]

• Để làm việc với tập dữ liệu hình ảnh, hãy cài đặt tính năng Hình ảnh:

pip install datasets[vision]

Bên cạnh Bộ dữ liệu, hãy đảm bảo đã cài đặt khung máy học ưa thích của bạn:

```bash pip cài đặt ngọn đuốc ``` ``` bash pip cài đặt tensorflow ```

Âm thanh

Tập dữ liệu âm thanh được tải giống như tập dữ liệu văn bản. Tuy nhiên, tập dữ liệu âm thanh được xử lý tru
hơi khác một chút. Thay vì trình mã thông báo, bạn sẽ cần một trình trích xuất tính năng. Đầu vào âm thanh
yêu cầu lấy mẫu lại tốc độ lấy mẫu của nó để phù hợp với tốc độ lấy mẫu của mô hình được huấn luyện trước
sử dụng. Trong phần bắt đầu nhanh này, bạn sẽ chuẩn bị tập dữ liệu MInDS-14 để huấn luyện mô hình và ph
vấn đề ngân hàng mà khách hàng đang gặp phải.

1. Load the MInDS-14 dataset by providing the load_dataset() function with the dataset name,
dataset configuration (not all datasets will have a configuration), and a dataset split:

```
>>> from datasets import load_dataset, Audio

>>> dataset = load_dataset("PolyAI/minds14", "en-US", split="train")
```

2. Tiếp theo, tải mô hình Wav2Vec2 đã được huấn luyện trước và trình trích xuất tính năng tương ứng của nó
Thư viện máy biến áp. Hoàn toàn bình thường khi thấy cảnh báo sau khi bạn tải mô hình về một số
trọng số không được khởi tạo. Điều này được mong đợi vì bạn đang tải điểm kiểm tra mô hình này
để đào tạo với một nhiệm vụ khác.

```
>>> from transformers import AutoModelForAudioClassification, AutoFeatureExtractor

>>> model = AutoModelForAudioClassification.from_pretrained("facebook/wav2vec2-base")
>>> feature_extractor = AutoFeatureExtractor.from_pretrained("facebook/wav2vec2-base")
```

3. Thẻ dữ liệu MInDS-14 cho biết tốc độ lấy mẫu là 8kHz, nhưng mô hình Wav2Vec2
```

đã được huấn luyện trước ở tốc độ lấy mẫu 16kHZ. Bạn sẽ cần lấy mẫu lại cột âm thanh bằng the cast_column() function and Audio feature to match the model's sampling rate.

```
>>> dataset = dataset.cast_column("audio", Audio(sampling_rate=16000))
>>> dataset[0]["audio"]
<datasets.features._torchcodec.AudioDecoder object at 0x11642b6a0>
```

4. Tạo một hàm để xử lý trước mảng âm thanh bằng trình trích xuất đặc trưng, đồng thời cắt bớt và ghép các chuỗi vào các tensor hình chữ nhật gọn gàng. Điều quan trọng nhất cần nhớ là gọi mảng âm thanh trong bộ trích xuất đặc trưng vì mảng - tín hiệu giọng nói thực tế - là đầu vào mô hình.

Once you have a preprocessing function, use the map() function to speed up processing by áp dụng hàm cho các lô ví dụ trong tập dữ liệu.

```
>>> def preprocess_function(examples):
...audio_arrays = [x.get_all_samples().data for x in examples["audio"]]
...inputs = feature_extractor(
...âm thanh_mảng,
...sampling_rate=16000,
...padding=True,
...max_length=100000,
...truncation=True,
...)
...trả lại đầu vào

>>> dataset = dataset.map(preprocess_function, batched=True)
```

5. Use the rename_column() function to rename the intent_class column to labels , which is tên đầu vào dự kiến trong Wav2Vec2ForSequenceClassification:

```
>>> dataset = dataset.rename_column("intent_class", "labels")
```

6. Đặt định dạng tập dữ liệu theo khung học máy bạn đang sử dụng.

Use the [set_format()](/docs/datasets/v4.2.0/en/package_reference/main_classes#datasets.Dataset.set_format) function to set the dataset format to `torch` and

chỉ định các cột bạn muốn định dạng. Chức năng này áp dụng định dạng một cách nhanh chóng. Sau đó chuyển đổi sang thang đo PyTorch, gói tập dữ liệu vào [`torch.utils.data.DataLoader`](https://alband.github.io/doc_view/data.html?highlight=torch%20utils%20data%20dataloader#torch.utils.data.DataLoader):

```
>>> from torch.utils.data import DataLoader

>>> dataset.set_format(type="torch", columns=["input_values", "labels"])
>>> dataloader = DataLoader(dataset, batch_size=4)
```

Sử dụng phương thức prepare_tf_dataset từ Transformers để chuẩn bị tập dữ liệu tương thích với TensorFlow và sẵn sàng đào tạo/tinh chỉnh một mô hình khi nó bao bọc Bộ dữ liệu HuggingFace dưới dạng tf.data.Dataset with collation and batching, so one can pass it directly to Keras methods like fit() without sửa đổi thêm.

```
>>> import tensorflow as tf

>>> tf_dataset = model.prepare_tf_dataset(
...tập dữ liệu,
...batch_size=4,
...shuffle=True,
... )
```

7. Bắt đầu đào tạo với khung học máy của bạn! Hãy xem Âm thanh của Transformers hướng dẫn phân loại để biết ví dụ chi tiết về cách đào tạo mô hình trên tập dữ liệu âm thanh.

Tầm nhìn

Tập dữ liệu hình ảnh được tải giống như tập dữ liệu văn bản. Tuy nhiên, thay vì mã thông báo, bạn sẽ cần mở trình trích xuất tính năng để xử lý trước tập dữ liệu. Áp dụng tăng cường dữ liệu cho hình ảnh là phổ biến trong thị giác máy tính để làm cho mô hình chống lại tình trạng quá khớp tốt hơn. Bạn được tự do sử dụng bất kỳ thư viện tăng cường dữ liệu nào bạn muốn và sau đó bạn có thể áp dụng các phần bổ sung v Bộ dữ liệu. Trong phần bắt đầu nhanh này, bạn sẽ tải tập dữ liệu Beans và chuẩn bị sẵn sàng cho mô hình huấn luyện và xác định bệnh từ hình ảnh chiếc lá.

1. Load the Beans dataset by providing the load_dataset() function with the dataset name and phân chia tập dữ liệu:

```
>>> from datasets import load_dataset, Image

>>> dataset = load_dataset("AI-Lab-Makerere/beans", split="train")
```

Hầu hết các mô hình hình ảnh đều hoạt động với hình ảnh RBG. Nếu tập dữ liệu của bạn chứa hình ảnh ở mộ mode, you can use the cast_column() function to set the mode to RGB:

```
>>> dataset = dataset.cast_column("image", Image(mode="RGB"))
```

Tập dữ liệu Beans chỉ chứa hình ảnh RGB nên bước này không cần thiết ở đây.

2. Now you can add some data augmentations with any library (Albumentations, imgaug, Kornia) you like. Here, you'll use torchvision to randomly change the color properties of an hình ảnh:

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

>>> jitter = Compose(
...[ColorJitter(brightness=0.5, hue=0.5), ToTensor()]
... )
```

3. Tạo một hàm để áp dụng phép biến đổi của bạn cho tập dữ liệu và tạo đầu vào mô hình: pixel_values .

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

4. Use the with_transform() function to apply the data augmentations on-the-fly:

```
>>> dataset = dataset.with_transform(transforms)
```

5. Đặt định dạng tập dữ liệu theo khung học máy bạn đang sử dụng.

Wrap the dataset in [`torch.utils.data.DataLoader`](https://alband.github.io/doc_view/data.html?highlight=torch%20utils%20data%20dataloader#torch.utils.data.DataLoader). You'll cũng cần tạo hàm đối chiếu để đối chiếu các mẫu thành từng đợt:

```
>>> from torch.utils.data import DataLoader

>>> def collate_fn(examples):
...images = []
...labels = []
...ví dụ trong ví dụ:
...images.append((example["pixel_values"]))
...labels.append(example["labels"])
None
...pixel_values = torch.stack(images)
...labels = torch.tensor(labels)
...return {"pixel_values": pixel_values, "labels": labels}
>>> dataloader = DataLoader(dataset, collate_fn=collate_fn, batch_size=4)
```

Sử dụng phương thức prepare_tf_dataset từ Transformers để chuẩn bị tập dữ liệu tương thích với TensorFlow và sẵn sàng đào tạo/tinh chỉnh một mô hình khi nó bao bọc Bộ dữ liệu HuggingFace dưới dạng tf.data.Dataset with collation and batching, so one can pass it directly to Keras methods like fit() without sửa đổi thêm.

Trước khi bắt đầu, hãy đảm bảo bạn có phiên bản cập nhật của albumentations và cv2 đã cài đặt:

```
cài đặt pip -U albumentations opencv-python
```

```
>>> import albumentations
>>> import numpy as np

>>> transform = albumentations.Compose([
...albumentations.RandomCrop(width=256, height=256),
...albumentations.HorizontalFlip(p=0.5),
...albumentations.RandomBrightnessContrast(p=0.2),
... ])

>>> def transforms(examples):
...examples["pixel_values"] = [
...transform(image=np.array(image))["image"] for image in examples["image"]
...]
...trả lại ví dụ

>>> dataset.set_transform(transforms)
>>> tf_dataset = model.prepare_tf_dataset(
...tập dữ liệu,
...batch_size=4,
...shuffle=True,
... )
```

6. Bắt đầu đào tạo với khung học máy của bạn! Hãy xem hình ảnh Máy biến áp
hướng dẫn phân loại để biết ví dụ chi tiết về cách đào tạo mô hình trên tập dữ liệu hình ảnh.

NLP

Văn bản cần được mã hóa thành các mã thông báo riêng lẻ bằng trình mã thông báo. Để bắt đầu nhanh, bạn
the Microsoft Research Paraphrase Corpus (MRPC) training dataset to train a model to
xác định xem một cặp câu có nghĩa giống nhau hay không.

1. Load the MRPC dataset by providing the load_dataset() function with the dataset name,
dataset configuration (not all datasets will have a configuration), and dataset split:

```
>>> from datasets import load_dataset

>>> dataset = load_dataset("nyu-mll/glue", "mrpc", split="train")
```

2. Tiếp theo, tải mô hình BERT đã được huấn luyện trước và mã thông báo tương ứng của nó từ Thư viện máy biến áp. Hoàn toàn bình thường khi thấy cảnh báo sau khi bạn tải mô hình về một số trọng số không được khởi tạo. Điều này được mong đợi vì bạn đang tải điểm kiểm tra mô hình này để đào tạo với một nhiệm vụ khác.

```
>>> from transformers import AutoModelForSequenceClassification, AutoTokenizer

>>> model = AutoModelForSequenceClassification.from_pretrained("bert-base-uncased")
>>> tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")
===PT-TF-SPLIT===
>>> from transformers import TFAutoModelForSequenceClassification, AutoTokenizer

>>> model = TFAutoModelForSequenceClassification.from_pretrained("bert-base-uncased")
>>> tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")
```

3. Tạo một hàm để mã hóa tập dữ liệu và bạn cũng nên cắt bớt và đệm văn bản vào tensor hình chữ nhật gọn gàng. Trình mã thông báo tạo ba cột mới trong tập dữ liệu: input_ids , token_type_ids và một chú ý_mask . Đây là những đầu vào của mô hình.

Use the map() function to speed up processing by applying your tokenization function to lô ví dụ trong tập dữ liệu:

```
>>> def encode(examples):
...return tokenizer(examples["sentence1"], examples["sentence2"], truncation=True, padding="m

>>> dataset = dataset.map(encode, batched=True)
>>> dataset[0]
{'sentence1': 'Amrozi accused his brother , whom he called " the witness " , of deliberately disto
'sentence2': 'Chỉ coi anh ta là "nhân chứng " , Amrozi buộc tội anh trai mình cố ý
'nhãn': 1,
'idx': 0,
'input_ids': [101,7277,2180,5303,4806,1117,1711,117,2292, 1119,1270,107,
'token_type_ids': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1
'attention_mask': [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
```

4. Đổi tên cột nhãn thành labels , đây là tên đầu vào dự kiến trong Phân loại BertForSequence:

```
>>> dataset = dataset.map(lambda examples: {"labels": examples["label"]}, batched=True)
```

5. Đặt định dạng tập dữ liệu theo khung học máy bạn đang sử dụng.

Use the [with_format()](/docs/datasets/v4.2.0/en/package_reference/main_classes#datasets.Dataset.with_format) function to set the dataset format to `torch` and chỉ định các cột bạn muốn định dạng. Chức năng này áp dụng định dạng một cách nhanh chóng. Sau đó chuyển đổi sang thang đo PyTorch, gói tập dữ liệu vào [`torch.utils.data.DataLoader`](https://alband.github.io/doc_view/data.html?highlight=torch%20utils%20data%20dataloader#torch.utils.data.DataLoader):

```
>>> import torch

>>> dataset = dataset.select_columns(["input_ids", "token_type_ids", "attention_mask", "labels"])
>>> dataset = dataset.with_format(type="torch")
>>> dataloader = torch.utils.data.DataLoader(dataset, batch_size=32)
```

Sử dụng phương thức prepare_tf_dataset từ Transformers để chuẩn bị tập dữ liệu tương thích với TensorFlow và sẵn sàng đào tạo/tinh chỉnh một mô hình khi nó bao bọc Bộ dữ liệu HuggingFace dưới dạng tf.data.Dataset with collation and batching, so one can pass it directly to Keras methods like fit() without sửa đổi thêm.

```
>>> import tensorflow as tf

>>> tf_dataset = model.prepare_tf_dataset(
...tập dữ liệu,
...batch_size=4,
...shuffle=True,
... )
```

6. Bắt đầu đào tạo với khung học máy của bạn! Hãy xem văn bản Transformers hướng dẫn phân loại để biết ví dụ chi tiết về cách đào tạo mô hình trên tập dữ liệu văn bản.

Tiếp theo là gì?

Việc này hoàn tất phần khởi động nhanh Bộ dữ liệu! Bạn có thể tải bất kỳ tập dữ liệu văn bản, âm thanh hoặ
một chức năng duy nhất và chuẩn bị sẵn sàng cho mô hình của bạn huấn luyện.

Đối với các bước tiếp theo của bạn, hãy xem hướng dẫn Cách thực hiện của chúng tôi và tìm hiểu cách thực h
như tải các định dạng tập dữ liệu khác nhau, căn chỉnh nhãn và truyền trực tuyến các tập dữ liệu lớn. Nếu bạ
muốn tìm hiểu thêm về Các khái niệm cốt lõi của bộ dữ liệu, hãy uống một tách cà phê và đọc
Hướng dẫn khái niệm!