Audio classification examples

The following examples showcase how to fine-tune `Wav2Vec2` for audio classification using PyTorch.

Speech recognition models that have been pretrained in unsupervised fashion on audio data alone, *e.g.* Wav2Vec2, HuBERT, XLSR-Wav2Vec2, have shown to require only very little annotated data to yield good performance on speech classification datasets.

Single-GPU

The following command shows how to fine-tune wav2vec2-base on the 🗣️ Keyword Spotting subset of the SUPERB dataset.

```
python run_audio_classification.py \
    --model_name_or_path facebook/wav2vec2-base \
    --dataset_name superb \
    --dataset_config_name ks \
    --output_dir wav2vec2-base-ft-keyword-spotting \
    --overwrite_output_dir \
    --remove_unused_columns False \
    --do_train \
    --do_eval \
    --fp16 \
    --learning_rate 3e-5 \
    --max_length_seconds 1 \
    --attention_mask False \
    --warmup_ratio 0.1 \
    --num_train_epochs 5 \
    --per_device_train_batch_size 32 \
    --gradient_accumulation_steps 4 \
    --per_device_eval_batch_size 32 \
    --dataloader_num_workers 4 \
    --logging_strategy steps \
    --logging_steps 10 \
    --evaluation_strategy epoch \
    --save_strategy epoch \
    --load_best_model_at_end True \
    --metric_for_best_model accuracy \
    --save_total_limit 3 \
    --seed 0 \
    --push_to_hub
```

On a single V100 GPU (16GB), this script should run in ~14 minutes and yield accuracy of **98.26%**.

👀 See the results here: anton-l/wav2vec2-base-ft-keyword-spotting

Multi-GPU

The following command shows how to fine-tune wav2vec2-base for 🌎 **Language Identification** on the CommonLanguage dataset.

```
python run_audio_classification.py \
    --model_name_or_path facebook/wav2vec2-base \
    --dataset_name common_language \
    --audio_column_name audio \
    --label_column_name language \
    --output_dir wav2vec2-base-lang-id \
    --overwrite_output_dir \
    --remove_unused_columns False \
    --do_train \
    --do_eval \
    --fp16 \
    --learning_rate 3e-5 \
    --max_length_seconds 16 \
    --attention_mask False \
    --warmup_ratio 0.1 \
    --num_train_epochs 10 \
    --per_device_train_batch_size 8 \
    --gradient_accumulation_steps 4 \
    --per_device_eval_batch_size 1 \
    --dataloader_num_workers 8 \
    --logging_strategy steps \
    --logging_steps 10 \
    --evaluation_strategy epoch \
    --save_strategy epoch \
    --load_best_model_at_end True \
    --metric_for_best_model accuracy \
    --save_total_limit 3 \
    --seed 0 \
    --push_to_hub
```

On 4 V100 GPUs (16GB), this script should run in ~1 hour and yield accuracy of **79.45%**.

👀 See the results here: anton-l/wav2vec2-base-lang-id

Sharing your model on 🤗 Hub

0. If you haven't already, sign up for a 🤗 account

1. Make sure you have `git-lfs` installed and git set up.

```
$ apt install git-lfs
```

2. Log in with your HuggingFace account credentials using `huggingface-cli`

```
$ huggingface-cli login
# ...follow the prompts
```

3. When running the script, pass the following arguments:

```
python run_audio_classification.py \
    --push_to_hub \
    --hub_model_id <username/model_id> \
    ...
```

Examples

The following table shows a couple of demonstration fine-tuning runs. It has been verified that the script works for the following datasets:

- SUPERB Keyword Spotting
- Common Language

Dataset	Pretrained Model	# transformer layers	Accuracy on eval	GPU setup	Training time	Fine-tuned Model & Logs
Keyword Spotting	ntu-spml/distilhubert	2	0.9706	1 V100 GPU	11min	here
Keyword Spotting	facebook/wav2vec2-base	12	0.9826	1 V100 GPU	14min	here
Keyword Spotting	facebook/hubert-base-ls960	12	0.9819	1 V100 GPU	14min	here
Keyword Spotting	asapp/sew-mid-100k	24	0.9757	1 V100 GPU	15min	here
Common Language	facebook/wav2vec2-base	12	0.7945	4 V100 GPUs	1h10m	here