Image classification examples

The following examples showcase how to fine-tune a `ViT` for image-classification using PyTorch.

Using datasets from 🤗 `datasets`

Here we show how to fine-tune a `ViT` on the beans dataset.

👀 See the results here: nateraw/vit-base-beans.

```
python run_image_classification.py \
    --dataset_name beans \
    --output_dir ./beans_outputs/ \
    --remove_unused_columns False \
    --do_train \
    --do_eval \
    --push_to_hub \
    --push_to_hub_model_id vit-base-beans \
    --learning_rate 2e-5 \
    --num_train_epochs 5 \
    --per_device_train_batch_size 8 \
    --per_device_eval_batch_size 8 \
    --logging_strategy steps \
    --logging_steps 10 \
    --evaluation_strategy epoch \
    --save_strategy epoch \
    --load_best_model_at_end True \
    --save_total_limit 3 \
    --seed 1337
```

Here we show how to fine-tune a `ViT` on the cats_vs_dogs dataset.

👀 See the results here: nateraw/vit-base-cats-vs-dogs.

```
python run_image_classification.py \
    --dataset_name cats_vs_dogs \
    --output_dir ./cats_vs_dogs_outputs/ \
    --remove_unused_columns False \
    --do_train \
    --do_eval \
    --push_to_hub \
    --push_to_hub_model_id vit-base-cats-vs-dogs \
    --fp16 True \
    --learning_rate 2e-4 \
    --num_train_epochs 5 \
    --per_device_train_batch_size 32 \
    --per_device_eval_batch_size 32 \
    --logging_strategy steps \
    --logging_steps 10 \
    --evaluation_strategy epoch \
```

```
    --save_strategy epoch \
    --load_best_model_at_end True \
    --save_total_limit 3 \
    --seed 1337
```

Using your own data

To use your own dataset, the training script expects the following directory structure:

```
root/dog/xxx.png
root/dog/xxy.png
root/dog/[...]/xxz.png

root/cat/123.png
root/cat/nsdf3.png
root/cat/[...]/asd932_.png
```

Once you've prepared your dataset, you can can run the script like this:

```
python run_image_classification.py \
    --dataset_name nateraw/image-folder \
    --train_dir <path-to-train-root> \
    --output_dir ./outputs/ \
    --remove_unused_columns False \
    --do_train \
    --do_eval
```

💡 The above will split the train dir into training and evaluation sets

- To control the split amount, use the `--train_val_split` flag.
- To provide your own validation split in its own directory, you can pass the `--validation_dir <path-to-val-root>` flag.

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
$ git config --global user.email "you@example.com"
$ git config --global user.name "Your Name"
```

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_image_classification.py \
    --push_to_hub \
    --push_to_hub_model_id <name-your-model> \
    ...
```