image-classification

September 20, 2023

```
[ ]: model_checkpoint = "microsoft/swin-tiny-patch4-window7-224" # pre-trained model
     batch_size = 32 # batch size for training and evaluation
```

```
[1]: !pip install -q datasets transformers
```

```
                                        519.6/519.6

     kB 5.4 MB/s eta 0:00:00
                                         7.6/7.6 MB
     26.2 MB/s eta 0:00:00
                                        115.3/115.3 kB
     15.0 MB/s eta 0:00:00
                                        194.1/194.1 kB
     22.5 MB/s eta 0:00:00
                                        134.8/134.8 kB
     16.3 MB/s eta 0:00:00
                                        294.9/294.9 kB
     32.3 MB/s eta 0:00:00
                                         7.8/7.8 MB
     68.9 MB/s eta 0:00:00
                                         1.3/1.3 MB
     76.3 MB/s eta 0:00:00
```

```
[2]: from huggingface_hub import notebook_login

     notebook_login()
```

```
     VBox(children=(HTML(value='<center> <img\nsrc=https://huggingface.co/front/
      ↪assets/huggingface_logo-noborder.sv…
```

```
[ ]: %%capture
     !sudo apt -qq install git-lfs
     !git config --global credential.helper store
```

```
[ ]: from transformers.utils import send_example_telemetry

     send_example_telemetry("image_classification_notebook", framework="pytorch")
```

0.0.1 Loading the dataset

```
[ ]: from datasets import load_dataset

     # load a custom dataset from local/remote files or folders using the␣
      ↪ImageFolder feature

     # option 1: local/remote files (supporting the following formats: tar, gzip,␣
      ↪zip, xz, rar, zstd)
     dataset = load_dataset("imagefolder", data_files="https://madm.dfki.de/files/
      ↪sentinel/EuroSAT.zip")

     # note that you can also provide several splits:
     # dataset = load_dataset("imagefolder", data_files={"train": ["path/to/file1",␣
      ↪"path/to/file2"], "test": ["path/to/file3", "path/to/file4"]})

     # note that you can push your dataset to the hub very easily (and reload␣
      ↪afterwards using load_dataset)!
     # dataset.push_to_hub("nielsr/eurosat")
     # dataset.push_to_hub("nielsr/eurosat", private=True)

     # option 2: local folder
     # dataset = load_dataset("imagefolder", data_dir="path_to_folder")

     # option 3: just load any existing dataset from the hub, like CIFAR-10,␣
      ↪FashionMNIST ...
     # dataset = load_dataset("cifar10")
```

```
Using custom data configuration default-0537267e6f812d56

Downloading and preparing dataset image_folder/default to /root/.cache/huggingfa
ce/datasets/image_folder/default-0537267e6f812d56/0.0.0/ee92df8e96c6907f3c851a98
7be3fd03d4b93b247e727b69a8e23ac94392a091…

Downloading data files: 0it [00:00, ?it/s]

Downloading data files:   0%|          | 0/1 [00:00<?, ?it/s]

Downloading data:   0%|          | 0.00/94.3M [00:00<?, ?B/s]

Extracting data files:   0%|          | 0/1 [00:00<?, ?it/s]

Generating train split: 0 examples [00:00, ? examples/s]

Dataset image_folder downloaded and prepared to /root/.cache/huggingface/dataset
s/image_folder/default-0537267e6f812d56/0.0.0/ee92df8e96c6907f3c851a987be3fd03d4
b93b247e727b69a8e23ac94392a091. Subsequent calls will reuse this data.

  0%|          | 0/1 [00:00<?, ?it/s]
```

[ ]:
```
from datasets import load_metric

metric = load_metric("accuracy")
```

```
Downloading builder script:   0%|          | 0.00/1.41k [00:00<?, ?B/s]
```

[ ]:
```
dataset
```

[ ]:
```
DatasetDict({
    train: Dataset({
        features: ['image', 'label'],
        num_rows: 27000
    })
})
```

[ ]:
```
example = dataset["train"][10]
example
```

[ ]:
```
{'image': <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=64x64 at
0x7FF2F6277B10>,
 'label': 2}
```

[ ]:
```
dataset["train"].features
```

[ ]:
```
{'image': Image(decode=True, id=None),
 'label': ClassLabel(num_classes=10, names=['AnnualCrop', 'Forest',
'HerbaceousVegetation', 'Highway', 'Industrial', 'Pasture', 'PermanentCrop',
'Residential', 'River', 'SeaLake'], id=None)}
```

[ ]:
```
example['image']
```

[ ]:

[ ]:
```
example['image'].resize((200, 200))
```

[ ]:

[ ]:
```
example['label']
```

[ ]:
```
2
```

[ ]:
```
dataset["train"].features["label"]
```

[ ]:
```
ClassLabel(num_classes=10, names=['AnnualCrop', 'Forest',
'HerbaceousVegetation', 'Highway', 'Industrial', 'Pasture', 'PermanentCrop',
'Residential', 'River', 'SeaLake'], id=None)
```

[ ]:
```
labels = dataset["train"].features["label"].names
label2id, id2label = dict(), dict()
for i, label in enumerate(labels):
    label2id[label] = i
    id2label[i] = label

id2label[2]
```

[ ]:
```
'HerbaceousVegetation'
```

0.0.2 Preprocessing the data

[ ]:
```
from transformers import AutoImageProcessor

image_processor  = AutoImageProcessor.from_pretrained(model_checkpoint)
image_processor
```

```
Downloading:   0%|          | 0.00/255 [00:00<?, ?B/s]
```

```
[ ]: ViTFeatureExtractor {
       "do_normalize": true,
       "do_resize": true,
       "feature_extractor_type": "ViTFeatureExtractor",
       "image_mean": [
         0.485,
         0.456,
         0.406
       ],
       "image_std": [
         0.229,
         0.224,
         0.225
       ],
       "resample": 3,
       "size": 224
     }
```

```
[ ]: from torchvision.transforms import (
         CenterCrop,
         Compose,
         Normalize,
         RandomHorizontalFlip,
         RandomResizedCrop,
         Resize,
         ToTensor,
     )

     normalize = Normalize(mean=image_processor.image_mean, std=image_processor.
      ↪image_std)
     if "height" in image_processor.size:
         size = (image_processor.size["height"], image_processor.size["width"])
         crop_size = size
         max_size = None
     elif "shortest_edge" in image_processor.size:
         size = image_processor.size["shortest_edge"]
         crop_size = (size, size)
         max_size = image_processor.size.get("longest_edge")

     train_transforms = Compose(
             [
                 RandomResizedCrop(crop_size),
                 RandomHorizontalFlip(),
                 ToTensor(),
                 normalize,
             ]
         )
```

```
val_transforms = Compose(
        [
            Resize(size),
            CenterCrop(crop_size),
            ToTensor(),
            normalize,
        ]
    )

def preprocess_train(example_batch):
    """Apply train_transforms across a batch."""
    example_batch["pixel_values"] = [
        train_transforms(image.convert("RGB")) for image in␣
 ↪example_batch["image"]
    ]
    return example_batch

def preprocess_val(example_batch):
    """Apply val_transforms across a batch."""
    example_batch["pixel_values"] = [val_transforms(image.convert("RGB")) for␣
 ↪image in example_batch["image"]]
    return example_batch
```

[ ]:
```
# split up training into training + validation
splits = dataset["train"].train_test_split(test_size=0.1)
train_ds = splits['train']
val_ds = splits['test']
```

[ ]:
```
train_ds.set_transform(preprocess_train)
val_ds.set_transform(preprocess_val)
```

[ ]:
```
train_ds[0]
```

[ ]:
```
{'image': <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=64x64 at
0x7FF2EFFB0D90>,
 'label': 9,
 'pixel_values': tensor([[[-0.3541, -0.3541, -0.3541,  …, -0.3712, -0.3712,
-0.3712],
          [-0.3541, -0.3541, -0.3541,  …, -0.3712, -0.3712, -0.3712],
          [-0.3541, -0.3541, -0.3541,  …, -0.3712, -0.3712, -0.3712],
          …,
          [-0.4397, -0.4397, -0.4397,  …, -0.4911, -0.4911, -0.4911],
          [-0.4397, -0.4397, -0.4397,  …, -0.4911, -0.4911, -0.4911],
          [-0.4397, -0.4397, -0.4397,  …, -0.4911, -0.4911, -0.4911]],

         [[-0.2500, -0.2500, -0.2500,  …, -0.2850, -0.2850, -0.2850],
          [-0.2500, -0.2500, -0.2500,  …, -0.2850, -0.2850, -0.2850],
          [-0.2500, -0.2500, -0.2500,  …, -0.2850, -0.2850, -0.2850],
          …,
          [-0.3550, -0.3550, -0.3550,  …, -0.4076, -0.4076, -0.4076],
          [-0.3550, -0.3550, -0.3550,  …, -0.4076, -0.4076, -0.4076],
          [-0.3550, -0.3550, -0.3550,  …, -0.4076, -0.4076, -0.4076]],

         [[ 0.1128,  0.1128,  0.1128,  …,  0.1651,  0.1651,  0.1651],
          [ 0.1128,  0.1128,  0.1128,  …,  0.1651,  0.1651,  0.1651],
          [ 0.1128,  0.1128,  0.1128,  …,  0.1651,  0.1651,  0.1651],
          …,
          [ 0.0605,  0.0605,  0.0605,  …,  0.0082,  0.0082,  0.0082],
          [ 0.0605,  0.0605,  0.0605,  …,  0.0082,  0.0082,  0.0082],
          [ 0.0605,  0.0605,  0.0605,  …,  0.0082,  0.0082,  0.0082]]]])}
```

0.0.3 Training the model

[ ]:
```
from transformers import AutoModelForImageClassification, TrainingArguments,␣
 ↪Trainer

model = AutoModelForImageClassification.from_pretrained(
    model_checkpoint,
    label2id=label2id,
    id2label=id2label,
    ignore_mismatched_sizes = True, # provide this in case you're planning to␣
 ↪fine-tune an already fine-tuned checkpoint
)
```

```
Downloading:   0%|          | 0.00/70.1k [00:00<?, ?B/s]

Downloading:   0%|          | 0.00/108M [00:00<?, ?B/s]

/usr/local/lib/python3.7/dist-packages/torch/functional.py:445: UserWarning:
torch.meshgrid: in an upcoming release, it will be required to pass the indexing
argument. (Triggered internally at
../aten/src/ATen/native/TensorShape.cpp:2157.)
  return _VF.meshgrid(tensors, **kwargs)  # type: ignore[attr-defined]
Some weights of SwinForImageClassification were not initialized from the model
checkpoint at microsoft/swin-tiny-patch4-window7-224 and are newly initialized
because the shapes did not match:
- classifier.weight: found shape torch.Size([1000, 768]) in the checkpoint and
torch.Size([10, 768]) in the model instantiated
- classifier.bias: found shape torch.Size([1000]) in the checkpoint and
torch.Size([10]) in the model instantiated
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
```

[ ]:

```
model_name = model_checkpoint.split("/")[-1]

args = TrainingArguments(
    f"{model_name}-finetuned-eurosat",
    remove_unused_columns=False,
    evaluation_strategy = "epoch",
    save_strategy = "epoch",
    learning_rate=5e-5,
    per_device_train_batch_size=batch_size,
    gradient_accumulation_steps=4,
    per_device_eval_batch_size=batch_size,
    num_train_epochs=3,
    warmup_ratio=0.1,
    logging_steps=10,
    load_best_model_at_end=True,
    metric_for_best_model="accuracy",
    push_to_hub=True,
)
```

[ ]:

```
import numpy as np

# the compute_metrics function takes a Named Tuple as input:
# predictions, which are the logits of the model as Numpy arrays,
# and label_ids, which are the ground-truth labels as Numpy arrays.
def compute_metrics(eval_pred):
    """Computes accuracy on a batch of predictions"""
    predictions = np.argmax(eval_pred.predictions, axis=1)
    return metric.compute(predictions=predictions, references=eval_pred.
 ↪label_ids)
```

[ ]:

```
import torch

def collate_fn(examples):
    pixel_values = torch.stack([example["pixel_values"] for example in␣
 ↪examples])
    labels = torch.tensor([example["label"] for example in examples])
    return {"pixel_values": pixel_values, "labels": labels}
```

[ ]:

```
trainer = Trainer(
    model,
    args,
    train_dataset=train_ds,
    eval_dataset=val_ds,
    tokenizer=image_processor,
    compute_metrics=compute_metrics,
    data_collator=collate_fn,
)
```

```
Cloning https://huggingface.co/nielsr/swin-tiny-patch4-window7-224-finetuned-
eurosat into local empty directory.
```

[ ]:

```python
train_results = trainer.train()
# rest is optional but nice to have
trainer.save_model()
trainer.log_metrics("train", train_results.metrics)
trainer.save_metrics("train", train_results.metrics)
trainer.save_state()
```

```
/usr/local/lib/python3.7/dist-packages/transformers/optimization.py:309:
FutureWarning: This implementation of AdamW is deprecated and will be removed in
a future version. Use the PyTorch implementation torch.optim.AdamW instead, or
set `no_deprecation_warning=True` to disable this warning
  FutureWarning,
***** Running training *****
  Num examples = 24300
  Num Epochs = 3
  Instantaneous batch size per device = 32
  Total train batch size (w. parallel, distributed & accumulation) = 128
  Gradient Accumulation steps = 4
  Total optimization steps = 570
```

```
<IPython.core.display.HTML object>
```

```
***** Running Evaluation *****
  Num examples = 2700
  Batch size = 32
Saving model checkpoint to swin-tiny-patch4-window7-224-finetuned-
eurosat/checkpoint-190
Configuration saved in swin-tiny-patch4-window7-224-finetuned-
eurosat/checkpoint-190/config.json
Model weights saved in swin-tiny-patch4-window7-224-finetuned-
eurosat/checkpoint-190/pytorch_model.bin
Feature extractor saved in swin-tiny-patch4-window7-224-finetuned-
eurosat/checkpoint-190/preprocessor_config.json
Feature extractor saved in swin-tiny-patch4-window7-224-finetuned-
eurosat/preprocessor_config.json
***** Running Evaluation *****
  Num examples = 2700
  Batch size = 32
Saving model checkpoint to swin-tiny-patch4-window7-224-finetuned-
eurosat/checkpoint-380
Configuration saved in swin-tiny-patch4-window7-224-finetuned-
eurosat/checkpoint-380/config.json
Model weights saved in swin-tiny-patch4-window7-224-finetuned-
eurosat/checkpoint-380/pytorch_model.bin
Feature extractor saved in swin-tiny-patch4-window7-224-finetuned-
eurosat/checkpoint-380/preprocessor_config.json
```

```
Feature extractor saved in swin-tiny-patch4-window7-224-finetuned-
eurosat/preprocessor_config.json
***** Running Evaluation *****
  Num examples = 2700
  Batch size = 32
Saving model checkpoint to swin-tiny-patch4-window7-224-finetuned-
eurosat/checkpoint-570
Configuration saved in swin-tiny-patch4-window7-224-finetuned-
eurosat/checkpoint-570/config.json
Model weights saved in swin-tiny-patch4-window7-224-finetuned-
eurosat/checkpoint-570/pytorch_model.bin
Feature extractor saved in swin-tiny-patch4-window7-224-finetuned-
eurosat/checkpoint-570/preprocessor_config.json
Feature extractor saved in swin-tiny-patch4-window7-224-finetuned-
eurosat/preprocessor_config.json


Training completed. Do not forget to share your model on huggingface.co/models
=)


Loading best model from swin-tiny-patch4-window7-224-finetuned-
eurosat/checkpoint-570 (score: 0.9744444444444444).
Saving model checkpoint to swin-tiny-patch4-window7-224-finetuned-eurosat
Configuration saved in swin-tiny-patch4-window7-224-finetuned-
eurosat/config.json
Model weights saved in swin-tiny-patch4-window7-224-finetuned-
eurosat/pytorch_model.bin
Feature extractor saved in swin-tiny-patch4-window7-224-finetuned-
eurosat/preprocessor_config.json
Saving model checkpoint to swin-tiny-patch4-window7-224-finetuned-eurosat
Configuration saved in swin-tiny-patch4-window7-224-finetuned-
eurosat/config.json
Model weights saved in swin-tiny-patch4-window7-224-finetuned-
eurosat/pytorch_model.bin
Feature extractor saved in swin-tiny-patch4-window7-224-finetuned-
eurosat/preprocessor_config.json
Several commits (2) will be pushed upstream.
The progress bars may be unreliable.
```

```
Upload file pytorch_model.bin:   0%|          | 3.34k/105M [00:00<?, ?B/s]
```

```
Upload file runs/Apr12_08-48-13_9520b574893c/events.out.tfevents.1649753401.
 ↪9520b574893c.77.0:  24%|##4       …
```

```
To https://huggingface.co/nielsr/swin-tiny-patch4-window7-224-finetuned-eurosat
   b46a767..6d6b8dc  main -> main

To https://huggingface.co/nielsr/swin-tiny-patch4-window7-224-finetuned-eurosat
```

```
   6d6b8dc..25dd5d7  main -> main
```

```
***** train metrics *****
  epoch                    =          3.0
  total_flos               = 1687935228GF
  train_loss               =       0.3276
  train_runtime            =   0:16:13.91
  train_samples_per_second =       74.852
  train_steps_per_second   =        0.585
```

We can check with the `evaluate` method that our `Trainer` did reload the best model properly (if it was not the last one):

[ ]:
```
metrics = trainer.evaluate()
# some nice to haves:
trainer.log_metrics("eval", metrics)
trainer.save_metrics("eval", metrics)
```

```
***** Running Evaluation *****
  Num examples = 2700
  Batch size = 32
```

```
<IPython.core.display.HTML object>
```

```
***** eval metrics *****
  epoch                   =        3.0
  eval_accuracy           =     0.9744
  eval_loss               =     0.0664
  eval_runtime            = 0:00:16.12
  eval_samples_per_second =     167.48
  eval_steps_per_second   =      5.273
```

[ ]:
```
trainer.push_to_hub()
```

```
Saving model checkpoint to swin-tiny-patch4-window7-224-finetuned-eurosat
Configuration saved in swin-tiny-patch4-window7-224-finetuned-
eurosat/config.json
Model weights saved in swin-tiny-patch4-window7-224-finetuned-
eurosat/pytorch_model.bin
Feature extractor saved in swin-tiny-patch4-window7-224-finetuned-
eurosat/preprocessor_config.json
```

```
Upload file runs/Apr12_08-48-13_9520b574893c/events.out.tfevents.1649754586.
 ↪9520b574893c.77.2: 100%|##########…
```

```
To https://huggingface.co/nielsr/swin-tiny-patch4-window7-224-finetuned-eurosat
   25dd5d7..2164338  main -> main
```

[ ]:

```
'https://huggingface.co/nielsr/swin-tiny-patch4-window7-224-finetuned-
eurosat/commit/2164338db59d40004286bc65800bfa50561ecd3d'
```

0.1 Inference

[ ]:

```
from PIL import Image
import requests

url = 'https://huggingface.co/nielsr/convnext-tiny-finetuned-eurostat/resolve/
 ↪main/forest.png'
image = Image.open(requests.get(url, stream=True).raw)
image
```

[ ]:

Google Earth
Image © 2017 DigitalGlobe
© 2017 Google
Image © 2017 CNES / Airbus
N
5 km

[ ]:

```
from transformers import AutoModelForImageClassification, AutoImageProcessor

repo_name = "nielsr/swin-tiny-patch4-window7-224-finetuned-eurosat"

image_processor = AutoImageProcessor.from_pretrained(repo_name)
model = AutoModelForImageClassification.from_pretrained(repo_name)
```

```
https://huggingface.co/nielsr/swin-tiny-patch4-window7-224-finetuned-
eurosat/resolve/main/preprocessor_config.json not found in cache or
```

```
force_download set to True, downloading to
/root/.cache/huggingface/transformers/tmpqggthctf
```

```
Downloading:   0%|          | 0.00/240 [00:00<?, ?B/s]
```

```
storing https://huggingface.co/nielsr/swin-tiny-patch4-window7-224-finetuned-
eurosat/resolve/main/preprocessor_config.json in cache at /root/.cache/huggingfa
ce/transformers/7b742d61fc51f2ef5f81a75f80b26419c9f5bd86cc3022ed5784d09823f219f2
.e34548f8325ec440fcf4990d4a8dbbfd665397400e9a700766de032d2b45cf6b
creating metadata file for /root/.cache/huggingface/transformers/7b742d61fc51f2e
f5f81a75f80b26419c9f5bd86cc3022ed5784d09823f219f2.e34548f8325ec440fcf4990d4a8dbb
fd665397400e9a700766de032d2b45cf6b
loading feature extractor configuration file https://huggingface.co/nielsr/swin-
tiny-patch4-window7-224-finetuned-eurosat/resolve/main/preprocessor_config.json
from cache at /root/.cache/huggingface/transformers/7b742d61fc51f2ef5f81a75f80b2
6419c9f5bd86cc3022ed5784d09823f219f2.e34548f8325ec440fcf4990d4a8dbbfd665397400e9
a700766de032d2b45cf6b
Feature extractor ViTFeatureExtractor {
  "do_normalize": true,
  "do_resize": true,
  "feature_extractor_type": "ViTFeatureExtractor",
  "image_mean": [
    0.485,
    0.456,
    0.406
  ],
  "image_std": [
    0.229,
    0.224,
    0.225
  ],
  "resample": 3,
  "size": 224
}

https://huggingface.co/nielsr/swin-tiny-patch4-window7-224-finetuned-
eurosat/resolve/main/config.json not found in cache or force_download set to
True, downloading to /root/.cache/huggingface/transformers/tmpzdd89w3g
```

```
Downloading:   0%|          | 0.00/1.24k [00:00<?, ?B/s]
```

```
storing https://huggingface.co/nielsr/swin-tiny-patch4-window7-224-finetuned-
eurosat/resolve/main/config.json in cache at /root/.cache/huggingface/transforme
rs/83e4a1dea85e8e284e4da8ae1e3cf950c2c7e74d65a5a188049b3371fcd151bd.f1ed4852dd8f
4c3d0c565427607bc41fff51b58ac73a0970bec8456e5c64cea0
creating metadata file for /root/.cache/huggingface/transformers/83e4a1dea85e8e2
84e4da8ae1e3cf950c2c7e74d65a5a188049b3371fcd151bd.f1ed4852dd8f4c3d0c565427607bc4
1fff51b58ac73a0970bec8456e5c64cea0
loading configuration file https://huggingface.co/nielsr/swin-tiny-
patch4-window7-224-finetuned-eurosat/resolve/main/config.json from cache at /roo
```

```
t/.cache/huggingface/transformers/83e4a1dea85e8e284e4da8ae1e3cf950c2c7e74d65a5a1
88049b3371fcd151bd.f1ed4852dd8f4c3d0c565427607bc41fff51b58ac73a0970bec8456e5c64c
ea0
Model config SwinConfig {
  "_name_or_path": "nielsr/swin-tiny-patch4-window7-224-finetuned-eurosat",
  "architectures": [
    "SwinForImageClassification"
  ],
  "attention_probs_dropout_prob": 0.0,
  "depths": [
    2,
    2,
    6,
    2
  ],
  "drop_path_rate": 0.1,
  "embed_dim": 96,
  "encoder_stride": 32,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.0,
  "hidden_size": 768,
  "id2label": {
    "0": "AnnualCrop",
    "1": "Forest",
    "2": "HerbaceousVegetation",
    "3": "Highway",
    "4": "Industrial",
    "5": "Pasture",
    "6": "PermanentCrop",
    "7": "Residential",
    "8": "River",
    "9": "SeaLake"
  },
  "image_size": 224,
  "initializer_range": 0.02,
  "label2id": {
    "AnnualCrop": 0,
    "Forest": 1,
    "HerbaceousVegetation": 2,
    "Highway": 3,
    "Industrial": 4,
    "Pasture": 5,
    "PermanentCrop": 6,
    "Residential": 7,
    "River": 8,
    "SeaLake": 9
  },
  "layer_norm_eps": 1e-05,
  "mlp_ratio": 4.0,
  "model_type": "swin",
  "num_channels": 3,
  "num_heads": [
    3,
    6,
    12,
    24
  ],
  "num_layers": 4,
  "patch_size": 4,
  "path_norm": true,
  "problem_type": "single_label_classification",
  "qkv_bias": true,
  "torch_dtype": "float32",
  "transformers_version": "4.18.0",
  "use_absolute_embeddings": false,
  "window_size": 7
}
```

```
https://huggingface.co/nielsr/swin-tiny-patch4-window7-224-finetuned-
eurosat/resolve/main/pytorch_model.bin not found in cache or force_download set
to True, downloading to /root/.cache/huggingface/transformers/tmpkh0vdu53
```

```
Downloading:   0%|          | 0.00/105M [00:00<?, ?B/s]
```

```
storing https://huggingface.co/nielsr/swin-tiny-patch4-window7-224-finetuned-
eurosat/resolve/main/pytorch_model.bin in cache at /root/.cache/huggingface/tran
sformers/3daadbe0cabef18dc0e2232ae080d135a9d4ee6b1dc7675725ef38bedb990b81.818e63
819e125637bd8a94f43b6899d1552f0b45884f1c28c458a5cb55dfa9e5
creating metadata file for /root/.cache/huggingface/transformers/3daadbe0cabef18
dc0e2232ae080d135a9d4ee6b1dc7675725ef38bedb990b81.818e63819e125637bd8a94f43b6899
d1552f0b45884f1c28c458a5cb55dfa9e5
loading weights file https://huggingface.co/nielsr/swin-tiny-
patch4-window7-224-finetuned-eurosat/resolve/main/pytorch_model.bin from cache
at /root/.cache/huggingface/transformers/3daadbe0cabef18dc0e2232ae080d135a9d4ee6
b1dc7675725ef38bedb990b81.818e63819e125637bd8a94f43b6899d1552f0b45884f1c28c458a5
cb55dfa9e5
All model checkpoint weights were used when initializing
SwinForImageClassification.

All the weights of SwinForImageClassification were initialized from the model
checkpoint at nielsr/swin-tiny-patch4-window7-224-finetuned-eurosat.
If your task is similar to the task the model of the checkpoint was trained on,
you can already use SwinForImageClassification for predictions without further
training.
```

[ ]:

```
# prepare image for the model
encoding = image_processor(image.convert("RGB"), return_tensors="pt")
print(encoding.pixel_values.shape)
```

```
torch.Size([1, 3, 224, 224])
```

[ ]:

```
import torch

# forward pass
with torch.no_grad():
    outputs = model(**encoding)
    logits = outputs.logits
```

[ ]:

```
predicted_class_idx = logits.argmax(-1).item()
print("Predicted class:", model.config.id2label[predicted_class_idx])
```

```
Predicted class: Forest
```