DSDE-Homework-1_Huggingface_image_classification

fine-tune ViT สำหรับทำ classification input เป็นใบของต้นถั่ว beans data set ด้วย Hugging Face library

โหลด datasets

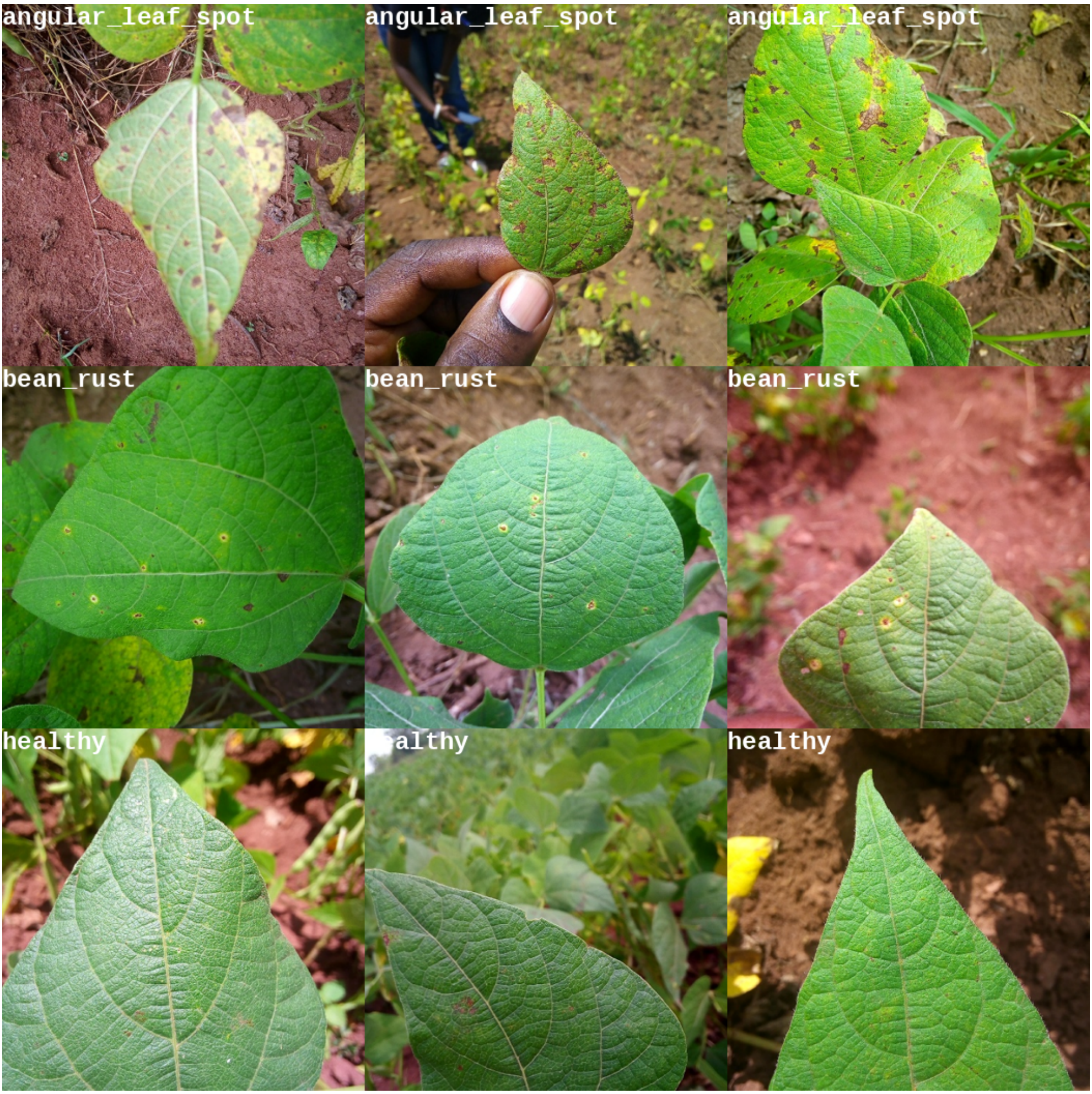
```
from datasets import load_dataset
ds = load_dataset('beans',)
ds
```

```
image = ds['train'][400]['image']
image
```

function สำหรับดู data ใน dataset

```
from transformers.utils.dummy_vision_objects import ImageGPTFeatureExtractor
import random
from PIL import ImageDraw, ImageFont, Image

def show_examples(ds, seed: int = 1234, examples_per_class: int = 3, size=(350, 350)):

    w, h = size
    labels = ds['train'].features['labels'].names
    grid = Image.new('RGB', size=(examples_per_class * w, len(labels) * h))
    draw = ImageDraw.Draw(grid)
    font = ImageFont.truetype("/usr/share/fonts/truetype/liberation/LiberationMono-Bold.ttf", 24)

    for label_id, label in enumerate(labels):

        # Filter the dataset by a single label, shuffle it, and grab a few samples
        ds_slice = ds['train'].filter(lambda ex: ex['labels'] ==
label_id).shuffle(seed).select(range(examples_per_class))

        # Plot this label's examples along a row
        for i, example in enumerate(ds_slice):
            image = example['image']
            idx = examples_per_class * label_id + i
            box = (idx % examples_per_class * w, idx // examples_per_class * h)
            grid.paste(image.resize(size), box=box)
            draw.text(box, label, (255, 255, 255), font=font)

    return grid
```

```
show_examples(ds, seed=random.randint(0, 1337), examples_per_class=3)
```



โหลด pretrained model

```
from transformers import ViTImageProcessor
model_name_or_path = 'google/vit-base-patch16-224'
feature_extractor = ViTImageProcessor.from_pretrained(model_name_or_path)
feature_extractor
```


```
ViTImageProcessor {
  "do_normalize": true,
  "do_rescale": true,
  "do_resize": true,
  "image_mean": [
    0.5,
    0.5,
    0.5
  ],
  "image_processor_type": "ViTImageProcessor",
```

```
  "image_std": [
    0.5,
    0.5,
    0.5
  ],
  "resample": 2,
  "rescale_factor": 0.00392156862745098,
  "size": {
    "height": 224,
    "width": 224
  }
}
```

function สำหรับ transform

```
def transform(example_batch):

# Take a list of PIL images and turn them to pixel values

inputs = feature_extractor([x for x in example_batch['image']], return_tensors='pt')

# Don't forget to include the labels!

inputs['labels'] = example_batch['labels']

return inputs



prepared_ds = ds.with_transform(transform)
```

```
import torch



def collate_fn(batch):

return { 'pixel_values': torch.stack([x['pixel_values'] for x in batch]),

'labels': torch.tensor([x['labels'] for x in batch])}
```

load accuracy metric สำหรับ evaluation

```
import numpy as np

from datasets import load_metric



metric = load_metric("accuracy")

def compute_metrics(p):

return metric.compute(predictions=np.argmax(p.predictions, axis=1), references=p.label_ids)
```

โหลด pretrain สำหรับ classification

```
from transformers import ViTForImageClassification



labels = ds['train'].features['labels'].names

model = ViTForImageClassification.from_pretrained('google/vit-base-patch16-224',

num_labels=len(labels),
```

```
ignore_mismatched_sizes=True)
```

ประกาศ Argument การ train

```
from transformers import TrainingArguments

training_args = TrainingArguments(
  output_dir="vit-base-beans-demo-v5",
  per_device_train_batch_size=16,
  per_device_eval_batch_size=16,
  evaluation_strategy="epoch",
  save_strategy="epoch",
  logging_strategy='epoch',
  logging_steps=1,
  num_train_epochs=10,
  optim='adamw_torch',
  learning_rate=1e-5,
  remove_unused_columns=False,
  report_to='tensorboard',
  load_best_model_at_end=True,
  metric_for_best_model="loss",
  seed=0
)
```

ประกาศ trainer

```
from transformers import Trainer

trainer = Trainer(
    model=model,
    args=training_args,
    data_collator=collate_fn,
    compute_metrics=compute_metrics,
    train_dataset=prepared_ds["train"],
    eval_dataset=prepared_ds["validation"],
    tokenizer=feature_extractor,
)
```

ทดลอง evaulate ก่อน fine tune

```
from sklearn.metrics import classification_report



test_predictions = trainer.predict(prepared_ds['test'])

# For each prediction, create the label with argmax

predictions = np.argmax(test_predictions[0], axis=1)

# Retrieve reference labels from test set

test_labels = np.array(prepared_ds["test"][:]["labels"])

print(classification_report(test_labels, predictions, target_names=['angular_leaf_spot', 'bean_rust', 'healthy'],
digits=4))
```

	precision	recall	f1-score	support
angular_leaf_spot	0.0000	0.0000	0.0000	43
bean_rust	0.3097	0.8140	0.4487	43
healthy	0.3333	0.0714	0.1176	42
accuracy			0.2969	128
macro avg	0.2144	0.2951	0.1888	128
weighted avg	0.2134	0.2969	0.1893	128

save result

```
train_results = trainer.train()

trainer.save_model()
trainer.log_metrics("train", train_results.metrics)

trainer.save_metrics("train", train_results.metrics)

trainer.save_state()
```

```
metrics = trainer.evaluate(prepared_ds['validation'])
trainer.log_metrics("eval", metrics)
trainer.save_metrics("eval", metrics)
```

```
***** eval metrics *****
  epoch                   =       10.0
  eval_accuracy           =     0.9925
  eval_loss               =     0.0242
  eval_runtime            = 0:00:03.01
  eval_samples_per_second =     44.081
  eval_steps_per_second   =      2.983
```

prediction

```
from sklearn.metrics import  classification_report

test_predictions = trainer.predict(prepared_ds['test'])
# For each prediction, create the label with argmax
predictions = np.argmax(test_predictions[0], axis=1)
# Retrieve reference labels from test set
test_labels = np.array(prepared_ds["test"][:]["labels"])
print(classification_report(test_labels, predictions, target_names=['angular_leaf_spot', 'bean_rust', 'healthy'],
digits=4))
```

ใช้ tensorboard

```
%load_ext tensorboard
%tensorboard --logdir '{"vit-base-beans-demo-v5"}'/runs
```

TensorBoard TIME SERIES SCALARS TEXT INACTIVE

Filter runs (regex)

Run

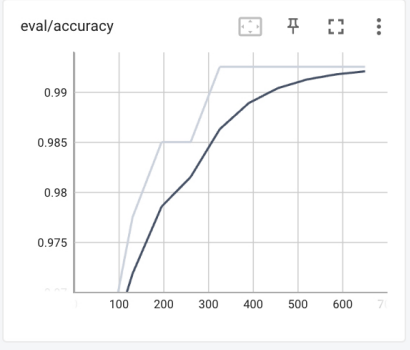
Sep02_08-13-09_3f014f08a587

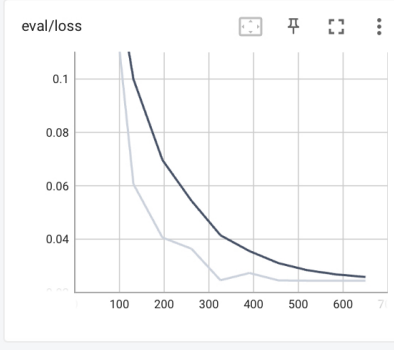
Filter tags (regex) All Scalars Image Histogram Settings

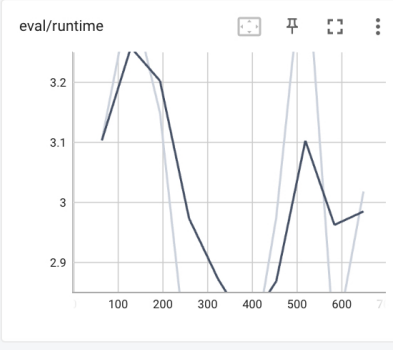
Pinned

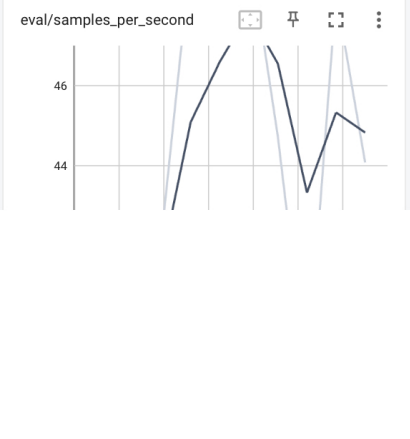
Pin cards for a quick view and comparison

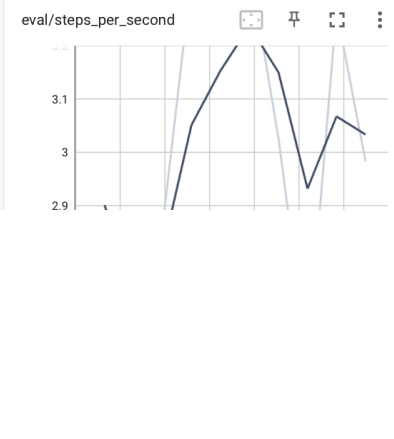
eval 5 cards











Settings

GENERAL

Horizontal Axis

Step

Enable step selection and data table (Scalars only)

Enable Range Selection

Link by step 65

Card Width

SCALARS

Smoothing 0.6

Tooltip sorting method

Alphabetical

Ignore outliers in chart scaling

Partition non-monotonic X axis

HISTOGRAMS

Mode

Offset