Image classification

Image classification datasets are used to train a model to classify an entire image. There are a wide variety of applications enabled by these datasets such as identifying endangered wildlife species or screening for disease in medical images. This guide will show you how to apply transformations to an image classification dataset.

Before you start, make sure you have up-to-date versions of `albumentations` and `cv2` installed:

```
pip install -U albumentations opencv-python
```

This guide uses the Beans dataset for identifying the type of bean plant disease based on an image of its leaf.

Load the dataset and take a look at an example:

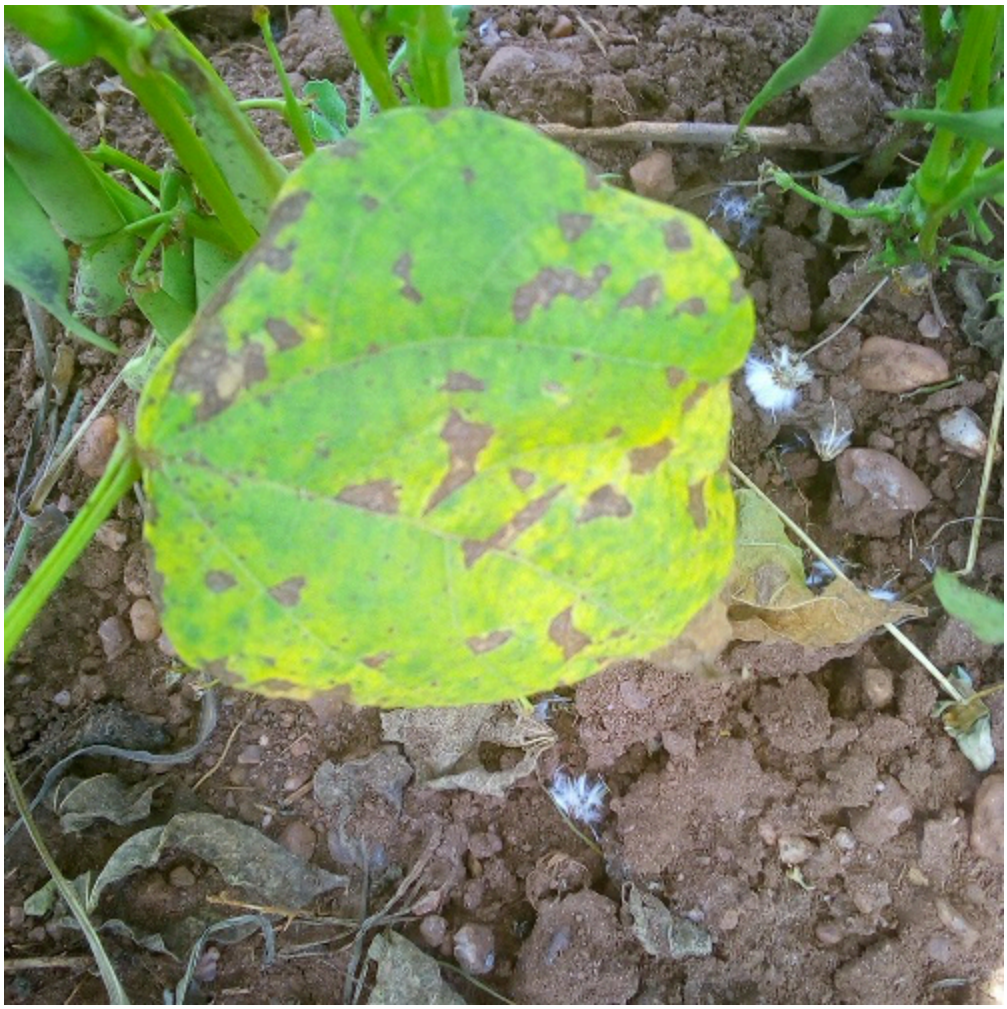
```
>>> from datasets import load_dataset

>>> dataset = load_dataset("AI-Lab-Makerere/beans")
>>> dataset["train"][10]
{'image': <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=500x500 at 0x7F8D2F4D7A10>,
 'image_file_path': '/root/.cache/huggingface/datasets/downloads/extracted/b0a21163f78769a2cf11f58
 'labels': 0}
```

The dataset has three fields:

- `image`: a PIL image object.
- `image_file_path`: the path to the image file.
- `labels`: the label or category of the image.

Next, check out an image:



Now apply some augmentations with `albumentations`. You'll randomly crop the image, flip it horizontally, and adjust its brightness.

```
>>> import cv2
>>> 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),
... ])
```

Create a function to apply the transformation to the images:

```
>>> def transforms(examples):
...     examples["pixel_values"] = [
...         transform(image=np.array(image))["image"] for image in examples["image"]
...     ]
...
...     return examples
```

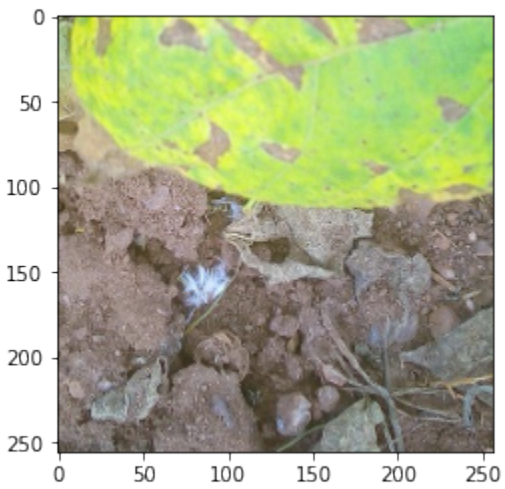
Use the set_transform() function to apply the transformation on-the-fly to batches of the dataset to consume less disk space:

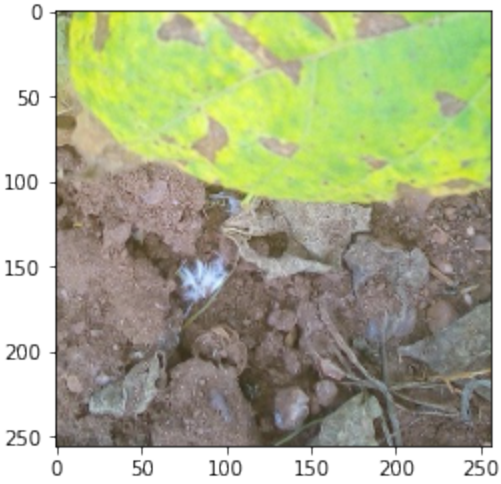
```
>>> dataset.set_transform(transforms)
```

You can verify the transformation worked by indexing into the `pixel_values` of the first example:

```
>>> import numpy as np
>>> import matplotlib.pyplot as plt

>>> img = dataset["train"][0]["pixel_values"]
>>> plt.imshow(img)
```





[!TIP]

Now that you know how to process a dataset for image classification, learn how to train an image classification model and use it for inference.