Data Augmentation

Solve the following exercises and upload your solutions to Moodle (unless specified otherwise) until the specified due date. Make sure to use the *exact filenames* that are specified for each individual exercise. Unless explicitly stated otherwise, you can assume correct user input and correct arguments. You are allowed to write additional functions, classes, etc. to improve readability and code quality.

Exercise 1 – Submission: `a6_ex1.py` 50 Points

Write a function

```
random_augmented_image(
    image: Image,
    image_size: Union[int, Sequence[int]],
    seed: int
) -> torch.Tensor
```

that applies a random chain of transforms on an image and returns a transformed image as a PyTorch tensor. The function's parameters are as follows:

- `image`: An input image loaded with Pillow (`PIL.Image`).
- `image_size`: The new size of the input image (see `torchvision.transforms.Resize`).
- `seed`: In each function call, a random chain of transforms is selected for application on the input image, where this seed is used for the reproducibility of the resulting output image.

The random chain of transforms consists of the following transformations:

1. `torchvision.transforms.Resize` (parameterized according to `image_size`, see above)
2. Exactly two transformations, chosen randomly from the following list of candidates:
   - `torchvision.transforms.RandomRotation`
   - `torchvision.transforms.RandomVerticalFlip`
   - `torchvision.transforms.RandomHorizontalFlip`
   - `torchvision.transforms.ColorJitter`
3. `torchvision.transforms.ToTensor`
4. `torch.nn.Dropout`

There are no specific requirements on the arguments and the order of transforms in the chain as long as their data types are compatible between two transforms and the data type of the function's output is a PyTorch tensor (`torch.Tensor`).

Example program execution:

```
if __name__ == "__main__":
    from matplotlib import pyplot as plt

    with Image.open("test_image.jpg") as image:
        transformed_image = random_augmented_image(image, image_size=300, seed=3)

        fig, axes = plt.subplots(1, 2)
        axes[0].imshow(image)
        axes[0].set_title("Original image")
        axes[1].imshow(transforms.functional.to_pil_image(transformed_image))
        axes[1].set_title("Transformed image")
        fig.tight_layout()
        plt.show()
```

Example output:

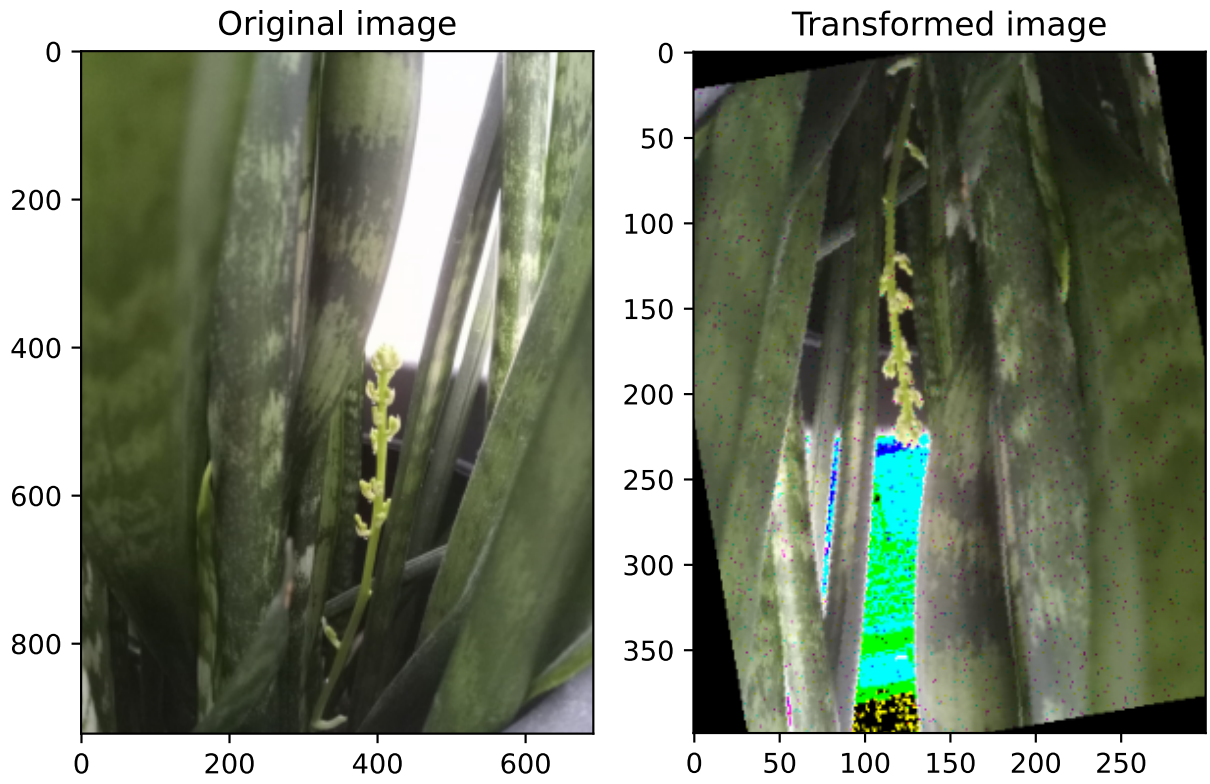

Figure 1: Visualization of the original and the transformed image by the above code.

Exercise 2 – Submission: `a6_ex2.py` 50 Points

Write the following two classes:

1. Class `ImageDataset` that extends `torch.utils.data.Dataset` to provide a data set of images. The following methods must be implemented:
   - `__init__(self, image_dir)`: The `image_dir` is a directory where all image file paths with the extension `".jpg"` should be searched for recursively, i.e., including subdirectories. The found file paths are collected in a list as absolute paths and sorted in an ascending order.
   - `__getitem__(self, index: int)`: Each individual image is retrieved from the data set via the `__getitem__` method through its corresponding index in the path list, and the returned data is a 2-tuple where the first entry is the PIL image and the second entry is the index.
   - `__len__(self)`: The length of the data set, i.e., the number of images.
2. Class `TransformedImageDataset` that extends `torch.utils.data.Dataset` to provide transformed images based on a provided `ImageDataset`. The following methods must be implemented:
   - `__init__(self, dataset: ImageDataset, image_size: Union[int, Sequence[int, int]])`: The `dataset` is an instance of class `ImageDataset` defined above. `image_size` is the same as in Exercise 1.
   - `__getitem__(self, index: int)`: Each individual image is retrieved from the `dataset` through its corresponding index. Afterwards, the image is transformed with the function `random_augmented_image` from Exercise 1. Use the index as seed and the `image_size` as specified in the `__init__` method. Analogous to `ImageDataset`, the method must return a 2-tuple where the first entry is the transformed PyTorch tensor image and the second entry is the index.
   - `__len__(self)`: The length of the data set, i.e., the number of transformed images (which is equal to the number of images in the provided `dataset`).

Example program execution:

```
if __name__ == "__main__":
    from matplotlib import pyplot as plt

    imgs = ImageDataset(image_dir="images")
    transformed_imgs = TransformedImageDataset(imgs, image_size=300)

    for (original_img, index), (transformed_img, _) in zip(imgs, transformed_imgs):
        fig, axes = plt.subplots(1, 2)
        axes[0].imshow(original_img)
        axes[0].set_title("Original image")
        axes[1].imshow(transforms.functional.to_pil_image(transformed_img))
        axes[1].set_title("Transformed image")
        fig.suptitle(f"Image {index}")
        fig.tight_layout()
        plt.show()
```

Example output (assuming some images in the provided directory):

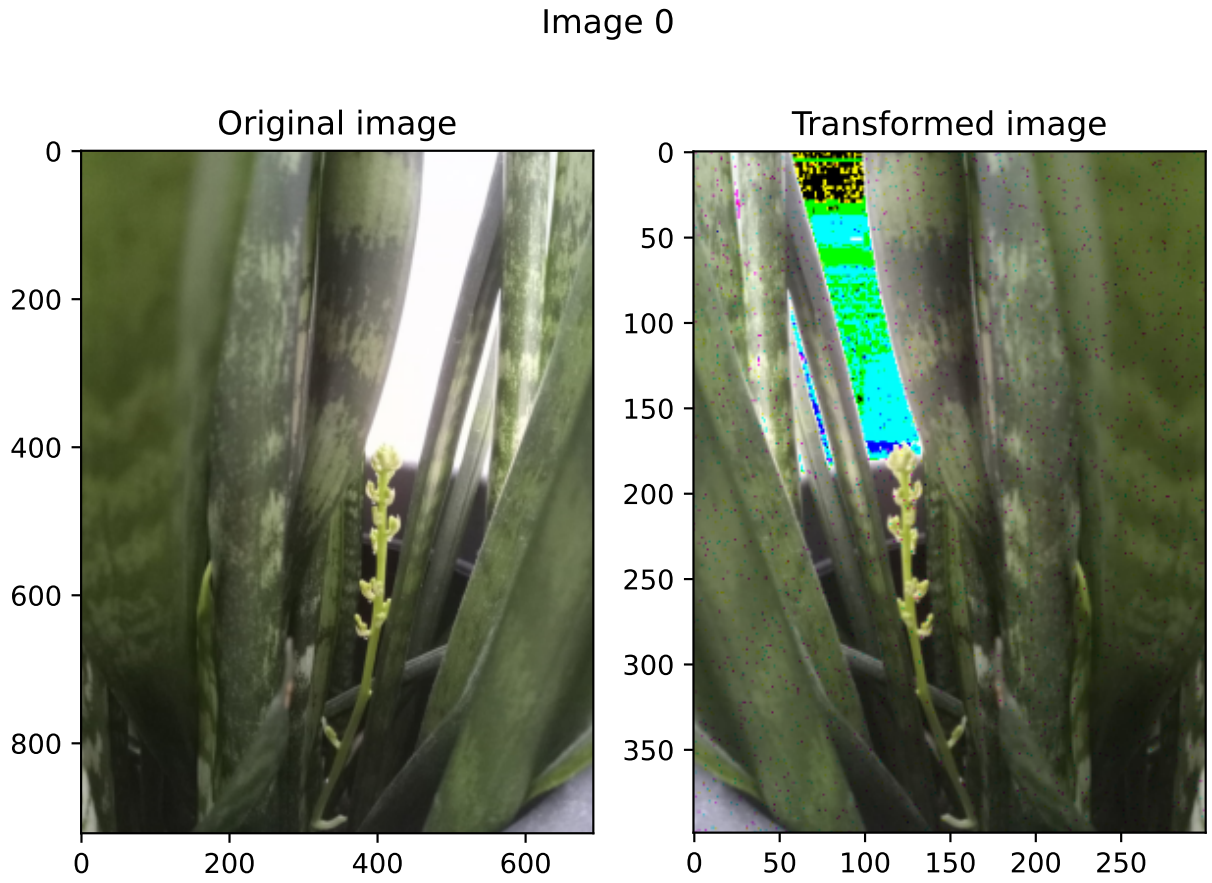

Figure 2: Visualization of the first image transformation by the above code.