pytorch

June 9, 2024

1 Data Augmentation Techniques With PyTorch

[13]:
```
import torchvision.transforms.functional as TF
from PIL import Image
import os
```

1.1 Defining the Augmentations

[14]:
```
def apply_transformations(image):
    # Horizontal and vertical flips
    flipped_image_x = TF.hflip(image)
    flipped_image_y = TF.vflip(image)

    # Saturation adjustments
    saturated_image_1 = TF.adjust_saturation(image, 0.5)
    saturated_image_2 = TF.adjust_saturation(image, 1.5)

    # Brightness adjustments
    brightened_image_1 = TF.adjust_brightness(image, 0.5)
    brightened_image_2 = TF.adjust_brightness(image, 2.0)

    # Contrast adjustments
    contrast_image_1 = TF.adjust_contrast(image, 0.5)
    contrast_image_2 = TF.adjust_contrast(image, 2.0)

    # Shear transformations using torchvision.transforms.functional.affine
    translated_image_10_20 = TF.affine(image, angle=0, translate=(10, 20),␣
↪scale=1.0, shear=(0, 0))
    translated_image_30_50 = TF.affine(image, angle=0, translate=(30, 50),␣
↪scale=1.0, shear=(0, 0))

    # Rotations
    rotated_image_90 = TF.rotate(image, 90)
    rotated_image_180 = TF.rotate(image, 180)

    return {
        'flipped_horizontal': flipped_image_x,
```

```
        'flipped_vertical': flipped_image_y,
        'saturated_0.5': saturated_image_1,
        'saturated_1.5': saturated_image_2,
        'brightened_0.5': brightened_image_1,
        'brightened_2.0': brightened_image_2,
        'contrast_0.5': contrast_image_1,
        'contrast_2.0': contrast_image_2,
        'translated_10_20': translated_image_10_20,
        'translated_30_50': translated_image_30_50,
        'rotated_90': rotated_image_90,
        'rotated_180': rotated_image_180
    }
```

1.2 Applying Filters

[15]:
```
def save_image(image, original_name, suffix, output_dir):
    base_name, ext = os.path.splitext(original_name)
    new_name = f"{base_name}_{suffix}{ext}"
    image.save(os.path.join(output_dir, new_name))

input_dir = '../images'
output_dir = 'pytorch_augmented_images'

image_files = [f for f in os.listdir(input_dir) if f.endswith(('.jpeg'))]

for image_file in image_files:
    image_path = os.path.join(input_dir, image_file)
    image = Image.open(image_path)

    transformed_images = apply_transformations(image)

    for suffix, img in transformed_images.items():
        save_image(img, image_file, suffix, output_dir)
```