Brightness

```
import os
import random
from PIL import Image, ImageEnhance
from tqdm import tqdm

def adjust_brightness(input_folder, output_folder):
    """
    Adjusts the brightness of all images in the input folder and saves
them to the output folder.

    Parameters:
        input_folder (str): Path to the folder containing input
images.
        output_folder (str): Path to the folder to save the processed
images.
    """
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    # Iterate over files in the input folder
    for filename in tqdm(os.listdir(input_folder), desc="Processing
Images (Brightness Adjustment)"):
        input_path = os.path.join(input_folder, filename)

        print(input_path)

        # Ensure the file is a valid image
        if os.path.isfile(input_path) and
filename.lower().endswith(('png', 'jpg', 'jpeg', 'bmp', 'gif')):
            try:
                with Image.open(input_path) as img:
                    # Generate a random brightness factor between 0.7
and 1.3
                    brightness_factor = random.uniform(0.7, 1.3)

                    # Enhance the image brightness
                    enhancer = ImageEnhance.Brightness(img)
                    img_enhanced = enhancer.enhance(brightness_factor)

                    # Save the processed image to the output folder
                    output_path = os.path.join(output_folder,
filename)

                    img_enhanced.save(output_path)
            except Exception as e:
                print(f"Failed to process {filename}: {e}")
```

```
# Example usage
input_folder = "/kaggle/input/gtsrb-german-traffic-sign/Test"
output_folder = "/kaggle/working/brightness_images"
adjust_brightness(input_folder, output_folder)
```

Motion Blur

```
import os
import random
from PIL import Image, ImageFilter
from tqdm import tqdm

def apply_motion_blur(input_folder, output_folder):
    """
    Applies a mild motion blur to all images in the input folder and
saves them to the output folder.

    Parameters:
        input_folder (str): Path to the folder containing input
images.
        output_folder (str): Path to the folder to save the processed
images.
    """
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    files = [f for f in os.listdir(input_folder) if
os.path.isfile(os.path.join(input_folder, f)) and
f.lower().endswith(('png', 'jpg', 'jpeg', 'bmp', 'gif'))]

    for filename in tqdm(files, desc="Processing Images (Motion
Blur)"):
        input_path = os.path.join(input_folder, filename)

        try:
            with Image.open(input_path) as img:
                # Generate a smaller blur radius between 1 and 3 for a
more subtle effect
                blur_radius = random.uniform(0.5, 3.0)

                # Apply Gaussian blur for a mild motion blur effect
                img_blurred =
img.filter(ImageFilter.GaussianBlur(blur_radius))

                # Save the processed image to the output folder
                output_path = os.path.join(output_folder, filename)
                img_blurred.save(output_path)
```

```
        except Exception as e:
            print(f"Failed to process {filename}: {e}")

# Example usage
input_folder = "/kaggle/input/gtsrb-german-traffic-sign/Test"
output_folder = "/kaggle/working/motion_blur_images"
apply_motion_blur(input_folder, output_folder)
```

Rain

```
import os
import random
from PIL import Image, ImageDraw, ImageFilter
from tqdm import tqdm

def add_rain_effect(input_folder, output_folder):
    """
    Adds a rainy effect to all images in the input folder and saves
them to the output folder.

    Parameters:
        input_folder (str): Path to the folder containing input
images.
        output_folder (str): Path to the folder to save the processed
images.
    """
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    files = [f for f in os.listdir(input_folder) if
os.path.isfile(os.path.join(input_folder, f)) and
f.lower().endswith(('png', 'jpg', 'jpeg', 'bmp', 'gif'))]

    for filename in tqdm(files, desc="Processing Images (Rain
Effect)"):
        input_path = os.path.join(input_folder, filename)

        try:
            with Image.open(input_path) as img:
                # Create a new image to draw rain streaks
                width, height = img.size
                rain_layer = Image.new("RGBA", img.size, (0, 0, 0, 0))
                draw = ImageDraw.Draw(rain_layer)

                # Add random rain streaks
                for _ in range(100):  # Adjust number of raindrops for
density
                    x_start = random.randint(0, width)
```

```
                    y_start = random.randint(0, height)
                    x_end = x_start + random.randint(-5, 5)
                    y_end = y_start + random.randint(10, 20)
                    opacity = random.randint(50, 100)  # Raindrop
transparency
                    draw.line((x_start, y_start, x_end, y_end),
fill=(200, 200, 255, opacity), width=1)

                # Blur the rain streaks to make them look natural
                rain_layer =
rain_layer.filter(ImageFilter.GaussianBlur(radius=0.5))

                # Composite the rain layer onto the original image
                img_with_rain =
Image.alpha_composite(img.convert("RGBA"), rain_layer)

                # Save the processed image to the output folder
                output_path = os.path.join(output_folder, filename)
                img_with_rain.convert("RGB").save(output_path)
        except Exception as e:
            print(f"Failed to process {filename}: {e}")

# Example usage
input_folder = "/kaggle/input/gtsrb-german-traffic-sign/Test"
output_folder = "/kaggle/working/rain"
add_rain_effect(input_folder, output_folder)
```

Snow

```
import os
import random
from PIL import Image, ImageDraw, ImageFilter
from tqdm import tqdm

def add_snow_effect(input_folder, output_folder):
    """
    Adds a snowy effect to all images in the input folder and saves
them to the output folder.

    Parameters:
        input_folder (str): Path to the folder containing input
images.
        output_folder (str): Path to the folder to save the processed
images.
    """
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)
```

```
    files = [f for f in os.listdir(input_folder) if
os.path.isfile(os.path.join(input_folder, f)) and
f.lower().endswith(('png', 'jpg', 'jpeg', 'bmp', 'gif'))]

    for filename in tqdm(files, desc="Processing Images (Snow
Effect)"):
        input_path = os.path.join(input_folder, filename)

        try:
            with Image.open(input_path) as img:
                # Create a new image to draw snowflakes
                width, height = img.size
                snow_layer = Image.new("RGBA", img.size, (255, 255,
255, 0))
                draw = ImageDraw.Draw(snow_layer)

                # Add random snowflakes
                for _ in range(500):  # Adjust number of snowflakes
for density
                    x = random.randint(0, width)
                    y = random.randint(0, height)
                    radius = random.randint(1, 3)  # Snowflake size
                    opacity = random.randint(150, 255)  # Snowflake
transparency
                    draw.ellipse(
                        (x - radius, y - radius, x + radius, y +
radius),
                        fill=(255, 255, 255, opacity)
                    )

                # Slight blur to make snowflakes more natural
                snow_layer =
snow_layer.filter(ImageFilter.GaussianBlur(radius=0.5))

                # Composite the snow layer onto the original image
                img_with_snow =
Image.alpha_composite(img.convert("RGBA"), snow_layer)

                # Save the processed image to the output folder
                output_path = os.path.join(output_folder, filename)
                img_with_snow.convert("RGB").save(output_path)
        except Exception as e:
            print(f"Failed to process {filename}: {e}")

# Example usage
input_folder = "/kaggle/input/gtsrb-german-traffic-sign/Test"
output_folder = "/kaggle/working/snow"
add_snow_effect(input_folder, output_folder)
```

Angle and rotation

```
import os
import random
from PIL import Image, ImageDraw, ImageFilter
from tqdm import tqdm

def addRotation(input_folder, output_folder):
    """
    Adds a snowy effect to all images in the input folder and saves
them to the output folder.

    Parameters:
        input_folder (str): Path to the folder containing input
images.
        output_folder (str): Path to the folder to save the processed
images.
    """
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    files = [f for f in os.listdir(input_folder) if
os.path.isfile(os.path.join(input_folder, f)) and
f.lower().endswith(('png', 'jpg', 'jpeg', 'bmp', 'gif'))]

    for filename in tqdm(files, desc="Processing Images (Snow
Effect)"):
        input_path = os.path.join(input_folder, filename)

        try:
            with Image.open(input_path) as img:
                random_angle = random.uniform(-45, 45)
                rotated_image = img.rotate(random_angle,
resample=Image.BICUBIC, expand=True)
                # Save the processed image to the output folder
                output_path = os.path.join(output_folder, filename)
                rotated_image.save(output_path)
        except Exception as e:
            print(f"Failed to process {filename}: {e}")

# Example usage
input_folder = "/kaggle/input/gtsrb-german-traffic-sign/Test"
output_folder = "/kaggle/working/rotate"
addRotation(input_folder, output_folder)
```