

Generative AI Journal

Introduction

For our project we all used chat GPT 4 and github co-pilot. We opted for these models over Gemini since we all already have paid subscriptions with OpenAI and it is more convenient to stick with what we have then change. As for github co-pilot, as students we get access to it for free, so it is a no brainer to make use of it.

Ethical Considerations

A few of the ethical considerations of using generative AI in an academic project are:

Privacy

The use of generative AI requires access to substantial amounts of data, some of which may be sensitive or personal. Ensuring that the data used is anonymized and handled with strict confidentiality is paramount to avoid privacy violations. Additionally, the transparency of the AI models used and the traceability of their outputs are crucial for maintaining ethical standards. This involves clearly documenting the AI's role in the project and ensuring that all AI-generated content is appropriately credited and verifiable.

Academic Integrity

When doing an academic project one must first ask himself the following question. What am I meant to learn through the process of this project. We were able to come up with 2 answers to that project.

1. To understand the core concepts and theory being implemented in the project.
2. To experience the trial and error process of implementing and developing a certain data structure/project.

For this project we felt like it was a bit of both. Thus we deemed it ok to use generative ai as a personal tutor to explain the core concepts behind some of the data augmentation techniques, aswell as having it correct our implementations once we had tried them at least once ourselves.

Methodology

We made use of generative AI at 2 key stages of our project.

Before attempting to answer a question we would show chat GPT what was being asked of us and request a personalised lesson explaining what we had to do and the key concepts behind each question.

Next after attempting to tackle the question ourselves, we would feed chat gpt our answers and ask him to optimize and correct them. This process allowed us to experience the trial and error loop while still obtaining finely optimized answers to each question.

Prompts and Responses

Prompt

Explain how to augment an image's brightness from base principles using OpenCV and NumPy

Response

Augmenting an image's brightness using OpenCV and NumPy involves manipulating the pixel values of the image. The process can be broken down into the following steps:

1. **Read the image:** Use OpenCV to load the image into a NumPy array.
2. **Convert to desired color space** (optional): If working with a color image, converting to a color space where brightness is easier to manipulate (like HSV) might be useful.
3. **Adjust brightness:** Modify the pixel values to increase or decrease the brightness.
4. **Clip values:** Ensure that the pixel values remain within the valid range (0 to 255).
5. **Convert back to the original color space** (if converted earlier).
6. **Save or display the image:** Use OpenCV to save or display the adjusted image.

Here's a step-by-step Python code example to illustrate this process:

Step-by-Step Code

1. Import Libraries:

```
import cv2
import numpy as np
```

2. Read the Image:

```
image = cv2.imread('path_to_your_image.jpg')
```

3. **Convert to HSV (if needed):** Converting the image to the HSV color space is useful because the V (value) channel directly represents brightness.

```
hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
```

4. **Adjust Brightness:** Brightness adjustment involves modifying the V channel.

```
# Split the HSV image into separate channels
h, s, v = cv2.split(hsv_image)

# Increase brightness by adding an offset to the V channel
brightness_offset = 50
v = cv2.add(v, brightness_offset)

# Clip the values to ensure they remain within [0, 255]
v = np.clip(v, 0, 255)
```

```
# Merge the channels back
adjusted_hsv_image = cv2.merge([h, s, v])
```

5. Convert Back to BGR:

```
adjusted_image = cv2.cvtColor(adjusted_hsv_image, cv2.COLOR_HSV2BGR)
```

6. Save or Display the Image:

```
# Save the adjusted image
cv2.imwrite('adjusted_image.jpg', adjusted_image)

# Display the original and adjusted images
cv2.imshow('Original Image', image)
cv2.imshow('Adjusted Image', adjusted_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Explanation

- **Reading the image:** `cv2.imread` loads the image into a NumPy array, where each pixel value is represented as an array of BGR (Blue, Green, Red) values.
- **Converting to HSV:** `cv2.cvtColor` converts the image from the BGR color space to the HSV (Hue, Saturation, Value) color space, which makes it easier to manipulate brightness.
- **Adjusting brightness:** By splitting the HSV image into its component channels, we can directly manipulate the V (value) channel to increase brightness. Adding an offset increases the brightness, and `np.clip` ensures the values stay within the valid range (0-255).
- **Merging channels:** After adjusting the brightness, we merge the channels back together and convert the image back to the BGR color space.
- **Saving or displaying the image:** Finally, we save the adjusted image or display it using OpenCV's `imshow`.

This approach provides a clear and effective way to augment the brightness of an image using basic principles with OpenCV and NumPy.

Prompt

I want to create a function that affects the brightness using some multiplier

Sure, you can create a function that adjusts the brightness of an image using a multiplier. This function will take an image and a multiplier as inputs, and it will increase or decrease the brightness of the image based on the given multiplier. Here's how you can do it:

Function to Adjust Brightness

```

import cv2
import numpy as np

def adjust_brightness(image, multiplier):
    """
    Adjust the brightness of an image using a multiplier.

    Parameters:
    - image: The input image (NumPy array).
    - multiplier: The brightness multiplier. Values > 1 will increase
    brightness,
                    values between 0 and 1 will decrease brightness.

    Returns:
    - adjusted_image: The brightness-adjusted image.
    """
    # Convert the image to HSV
    hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

    # Split the HSV image into separate channels
    h, s, v = cv2.split(hsv_image)

    # Apply the multiplier to the V (value) channel
    v = np.clip(v * multiplier, 0, 255).astype(np.uint8)

    # Merge the channels back
    adjusted_hsv_image = cv2.merge([h, s, v])

    # Convert back to BGR
    adjusted_image = cv2.cvtColor(adjusted_hsv_image,
    ``python
    cv2.COLOR_HSV2BGR)

    return adjusted_image

# Example usage
if __name__ == "__main__":
    # Read the image
    image = cv2.imread('path_to_your_image.jpg')

    # Adjust the brightness with a multiplier (e.g., 1.5 for 50% increase)
    multiplier = 1.5
    adjusted_image = adjust_brightness(image, multiplier)

    # Save or display the adjusted image
    cv2.imwrite('adjusted_image.jpg', adjusted_image)
    cv2.imshow('Original Image', image)
    cv2.imshow('Adjusted Image', adjusted_image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

```

Explanation

1. Function Definition:

- The function `adjust_brightness` takes an image and a multiplier as input.
- The image is converted to the HSV color space using `cv2.cvtColor`.
- The HSV image is split into its three channels: H (hue), S (saturation), and V (value/brightness).
- The V channel is multiplied by the given multiplier to adjust the brightness. Using `np.clip` ensures that the values stay within the valid range (0-255). The result is cast back to an unsigned 8-bit integer type using `astype(np.uint8)` to ensure compatibility with OpenCV functions.
- The modified channels are merged back together, and the image is converted back to the BGR color space.
- The adjusted image is returned.

2. Usage Example:

- The image is read from a file using `cv2.imread`.
- The `adjust_brightness` function is called with the image and a brightness multiplier (e.g., 1.5 to increase brightness by 50%).
- The adjusted image is saved to a file using `cv2.imwrite` and displayed using `cv2.imshow`.

This function allows you to easily adjust the brightness of an image by specifying a multiplier, providing a flexible and effective way to control image brightness.

Prompt

```
def adjust_brightness(image, brightness_factor):
    r, g, b = image[..., 0], image[..., 1], image[..., 2]
    hue, saturation, lightness = rgb_to_hsl(r, g, b)
    lightness *= brightness_factor
    lightness = np.clip(lightness, 0, 1)

    r, g, b = hsl_to_rgb(hue, saturation, lightness)
    new_pixels = np.stack([r, g, b], axis=-1)

    new_image = Image.fromarray(new_pixels, 'RGB')
    return new_image
```

Here is my implementation using hsl to rgb conversions, i want to adjust it to use open cv

Response

Sure, let's modify your implementation to use OpenCV for adjusting brightness by working directly with the HSV (Hue, Saturation, Value) color space. Here is the adjusted function:

Adjusted Brightness Function Using OpenCV

```

import cv2
import numpy as np

def adjust_brightness(image, brightness_factor):
    """
    Adjust the brightness of an image using a brightness factor.

    Parameters:
    - image: The input image (NumPy array).
    - brightness_factor: The brightness factor. Values > 1 will increase
    brightness,
                        values between 0 and 1 will decrease brightness.

    Returns:
    - adjusted_image: The brightness-adjusted image.
    """
    # Convert the image to HSV color space
    hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

    # Split the HSV image into separate channels
    h, s, v = cv2.split(hsv_image)

    # Apply the brightness factor to the V (value) channel
    v = np.clip(v * brightness_factor, 0, 255).astype(np.uint8)

    # Merge the channels back
    adjusted_hsv_image = cv2.merge([h, s, v])

    # Convert back to BGR color space
    adjusted_image = cv2.cvtColor(adjusted_hsv_image, cv2.COLOR_HSV2BGR)

    return adjusted_image

# Example usage
if __name__ == "__main__":
    # Read the image
    image = cv2.imread('path_to_your_image.jpg')

    # Adjust the brightness with a brightness factor (e.g., 1.5 for 50%
    increase)
    brightness_factor = 1.5
    adjusted_image = adjust_brightness(image, brightness_factor)

    # Save or display the adjusted image
    cv2.imwrite('adjusted_image.jpg', adjusted_image)
    cv2.imshow('Original Image', image)
    cv2.imshow('Adjusted Image', adjusted_image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

```

Explanation

1. Function Definition:

- The function `adjust_brightness` takes an image and a brightness factor as input.
- The image is converted to the HSV color space using `cv2.cvtColor` with the flag `cv2.COLOR_BGR2HSV`.
- The HSV image is split into its three channels: H (hue), S (saturation), and V (value/brightness) using `cv2.split`.
- The V channel is multiplied by the brightness factor to adjust the brightness. The resulting values are clipped to the valid range (0-255) using `np.clip` and then converted back to an unsigned 8-bit integer type.
- The modified H, S, and V channels are merged back together using `cv2.merge`.
- The merged HSV image is converted back to the BGR color space using `cv2.cvtColor` with the flag `cv2.COLOR_HSV2BGR`.
- The adjusted image is returned.

2. Usage Example:

- The image is read from a file using `cv2.imread`.
- The `adjust_brightness` function is called with the image and a brightness factor (e.g., 1.5 to increase brightness by 50%).
- The adjusted image is saved to a file using `cv2.imwrite` and displayed using `cv2.imshow`.

This approach leverages OpenCV's efficient color space conversion and manipulation functions to adjust the brightness of an image.

Further

This approach was used for all of the data augmentation techniques. Including all of the responses used would take us way over the 10 page limit for this document, thus we opted to show the process for one technique.

Improvements and Contributions

Chat GPT contributed to the following sections of our project:

1. Explanation: With AI we can ensure we are understanding what the assignment is asking of us.
2. Debugging: Using Gen-AI we were able to efficiently debug our work.
3. Optimizing: For the sections above, AI was used to optimize our final solution to make sure it ran as fast as possible.

Individual Reflection

Nick:

The use of generative AI significantly enhanced my understanding of the project requirements and provided clear guidance on how to execute tasks effectively. The ability to attach documents, code, screenshots, and other resources makes this tool highly versatile, essentially acting as a personal tutor with answers to almost any question. I was particularly impressed by its speed and accuracy, especially in recognising and processing screenshots and user interfaces.

My experience with this assignment has reinforced my belief that AI should be permitted in academic projects due to its powerful capabilities. While there is a concern that some students might misuse AI by merely prompting it for answers without further thought, I believe that learning to effectively utilise AI is a valuable skill. This assignment has cemented my view that AI can greatly benefit academic work when used responsibly and thoughtfully.

Saul

Reflecting on my experience using generative AI in my academic project, I found it both enlightening and transformative. AI's versatility in automating tasks and analysing large datasets was invaluable, allowing more time for critical thinking. I was surprised by the sophistication and accuracy of AI models, which exceeded my expectations. This experience highlighted the importance of understanding AI's limitations and using it to augment, not replace, human judgment.

My perspective on AI in academic projects has evolved from seeing it as a supplementary tool to recognising it as a transformative technology. It has the potential to reshape research methodologies, but it also requires careful ethical considerations and transparency. Overall, this experience has made me more optimistic about AI's role in driving academic innovation and discovery.

Luca

Reflecting on my use of generative AI in this project, I found it to be an immensely powerful tool. The AI's instant, detailed explanations and code snippets made complex tasks more approachable, and its proficiency in handling diverse questions and generating accurate responses made it a vital resource.

This AI-supported assignment has deepened my understanding of its potential in academics. While concerns about misuse exist, I believe responsible use of AI can greatly enrich learning. This project reinforced my belief that mastering AI tools is essential for future pursuits, offering efficiency and innovation.

AI's ability to automate tasks and provide advanced analytics allowed me to focus more on critical thinking and problem-solving. This experience has improved my technical skills and emphasised the importance of ethical AI use. Overall, I am optimistic about AI's role in education, seeing it as a catalyst for innovation and a valuable academic ally.