ARI2129 - Group Project 2024

Part B - Generative AI Journal

Name: **Andrea F. Lucas, Thomas Koppens & Sean D. Muscat**
Course: ARI2129 - Artificial Intelligence (AI)
ID No: 0279704L, 029204L & 0172004L

Distribution of Work - Generative AI Journal

Section	**Responsible Person(s)**
Introduction	Andrea F. Lucas & Sean D. Muscat
Ethical Considerations	Andrea F. Lucas
Methodology	Thomas Koppens
Section: Andrea Filiberto Lucas (Prompts, Responses, & Individual Reflection)	Andrea F. Lucas
Section: Thomas Koppens (Prompts, Responses, & Individual Reflection)	Thomas Koppens
Section: Sean David Muscat (Prompts, Responses, & Individual Reflection)	Sean D. Muscat
Improvements and Contributions	Sean D. Muscat
References	Andrea F. Lucas

---

1 Introduction

In the second section of our computer vision (CV) assignment, we investigate the dramatic impact of generative AI models on productivity and accuracy. We, Andrea Lucas, Sean Muscat, & Thomas Koppens utilized various generative AI models, carefully selecting each to maximize their unique capabilities.

GPT-4 and its latest iteration, GPT-4o, were at the forefront of our GenAI usage, due to their exceptional natural language processing (NLP) capabilities [1-3]. Their integration into our workflow was facilitated by their outstanding capacity to comprehend and generate code, as well as their accuracy in responses. These models served as the foundation for our code optimisation efforts, making comprehensive suggestions and changes that significantly streamlined our functions and increased overall efficiency. Furthermore, their advanced image analysis features proved invaluable as supplementary review systems for detecting anomalies in output images, providing insightful feedback to our analysis process.

In addition, we used ChatGPT's free 3.5 version to conduct code reviews and ensure project adherence. Its conversational interface enabled us to thoroughly review our code and programmes, ensuring they met the specified criteria.

Along with the GPT series, we used Google's Gemini 1.0 Pro [4-6], a generative AI tool that excels at writing tasks. Gemini's ability to generate documentation (within the.ipynb files) and ensure code clarity using in-line comments was outstanding. Despite using the free version, Gemini proved to be more than adequate for meeting our documentation requirements, seamlessly integrating into our workflow, and improving the comprehensibility of our project.

To maximise the benefits of both free and paid models while maintaining fairness, we used a multi-model approach, strategically combining Google Gemini [4], GPT-4, GPT-4o and GPT-3.5. This integration enabled us to capitalise on each model's strengths, improving technical accuracy and documentation efficiency while also strengthening the project's integrity. We not only succeeded in our CV assignment thanks to these collaborative efforts, but we also helped to advance the understanding and application of generative AI in real-world scenarios, as well as deepen our understanding on the topic at hand - Data Augmentation and its important use cases in CV [7].

---

2 Ethical Considerations

Incorporating generative AI into any project necessitates careful consideration of ethical implications, particularly those related to data bias and privacy. Addressing these ethical aspects was critical in our project, which used a variety of generative AI models, as mentioned in Section 1, to ensure responsible and fair technology use [8-9].

The risk of **data bias** is one of the most important ethical concerns when using GenAI [9-11]. Generative models are trained on large datasets, which may contain unintentional bias. These biases can be seen in the generated outputs, which reinforce stereotypes, discrimination, and even inequality. As a result, we continuously monitor model outputs for any signs of bias, quickly mitigating any problems that arise [9-10].

Privacy emerges as an important ethical consideration in the use of generative AI models, particularly in university assignments [12-14]. These models frequently process large amounts of data, including sensitive or confidential information. We took deliberate steps to protect data privacy in our project after recognising the possibility of such information being present in assignment prompts or datasets.

We additionally dealt with the ethical concern about potential unintended consequences from the use of generative AI. While these models have great potential for innovation and progress, they also carry risks like misinformation, manipulation, and misuse [15]. Throughout the project, we prioritised thorough validation and testing procedures to ensure the accuracy and reliability of model results. Furthermore, we thoroughly reviewed, customised, and refined the Gen AI output, actively reducing errors.

In conclusion, incorporating generative AI into projects necessitates addressing ethical concerns like data bias, privacy, and unintended consequences. Our project diligently addressed these issues by monitoring and mitigating biases in model outputs [8], protecting data privacy [12], and ensuring accuracy through rigorous validation and refinement procedures [15]. By prioritising ethical considerations [8, 10-12], we promoted responsible and equitable technology use, establishing an example for ethical AI deployment in a variety of contexts, within this assignment.

NB: *We also benefited from the inherent impartiality of the dataset provided by our lecture, Dr Dylan Seychell, which included images of inanimate objects from the Common Objects of a Traveling Scientist (COTS) dataset [16-17]. Given the absence of human subjects and personal data in this dataset, ethical concerns about privacy and bias were inherently limited.*

---

3 Methodology

When project members encountered difficulties with their part of the assignment, generative AI (GenAI) proved to be an invaluable resource. Learning new machine learning frameworks was one of the most challenging aspects. When the team needed to use frameworks they were not familiar with, GenAI was used to quickly grasp the concepts and functionalities. This included understanding how different modules within the frameworks functioned and how to effectively implement them in their section of the project. Furthermore, when the team needed to understand specific techniques or algorithms, GenAI provided in-depth explanations. This helped the team understand the inner workings of these techniques, ensuring that they were used correctly and efficiently.

GenAI was also useful in improving the project's documentation in part A and B of the assignment. In part A, GenAI assisted in the structuring of reports by advising on how to organise content logically and coherently. This included providing feedback on the document's flow and ensuring that all relevant sections were covered thoroughly. In part B, GenAI was used to check the report for quality. It helped to ensure that the information presented was accurate, relevant, and complete. Additionally, it checked that the narrative was consistent and easy to understand, corrected grammatical errors, improved sentence structures, and ensured correct punctuation throughout the document.

Integrating GenAI into the team's workflow was straightforward, with most interactions requiring only one step. A project member would ask GenAI a specific question. Its response would then dictate how the member moved forward. For example, if the AI recommended a different parameter setting in a machine learning model to improve performance, the team would make the change and monitor its effectiveness. Similarly, if the AI identified areas of the report that required improvement, the team would revise those sections accordingly.

In addition, GenAI was used to add inline comments to the Jupyter Notebooks used in the project. These comments added context and explanations to various code segments, making the notebooks more understandable and follow-able for anyone reviewing the work. Furthermore, the team fed images into GenAI, including visualisations of the Structural Similarity Index (SSIM) and the actual augmented images. The AI then analysed the images, providing feedback on their quality and relevance. This feedback was provided insights used for refining the project's visual components, ensuring that the images effectively communicated results.

Using GenAI in these ways allowed the team to efficiently overcome technical challenges, ensure high-quality documentation, and streamline their workflow. The AI served as both a technical advisor and an editorial assistant, offering support throughout the project's life cycle, thus improving the project deliverables' clarity and quality.

4 Section: Andrea Filiberto Lucas

4.1 Prompts & Responses

Prompt 1:

Please add in-line comments to:

```
def calculate_ssim(image1, image2):
    image1_np = tf.keras.preprocessing.image.img_to_array(image1).astype('float32') / 255.0
    image2_np = tf.keras.preprocessing.image.img_to_array(image2).astype('float32') / 255.0

    image1_gray = tf.image.rgb_to_grayscale(image1_np)
    image2_gray = tf.image.rgb_to_grayscale(image2_np)

    image1_gray_np = image1_gray.numpy().squeeze()
    image2_gray_np = image2_gray.numpy().squeeze()

    score, _ = ssim(image1_gray_np, image2_gray_np, full=True, data_range=image2_gray_np.max() - image2_gray_np.min())
    return score
```

Response 1 *(GPT 3.5)*:

Here is the function with added in-line comments:

```
def calculate_ssim(image1, image2):
    # Ensure images are NumPy arrays and convert to grayscale
    image1_np = tf.keras.preprocessing.image.img_to_array(image1).astype('float32') / 255.0
    image2_np = tf.keras.preprocessing.image.img_to_array(image2).astype('float32') / 255.0

    # Convert RGB images to grayscale
    image1_gray = tf.image.rgb_to_grayscale(image1_np)
    image2_gray = tf.image.rgb_to_grayscale(image2_np)

    # Convert grayscale TensorFlow tensors to NumPy arrays
    image1_gray_np = image1_gray.numpy().squeeze()
    image2_gray_np = image2_gray.numpy().squeeze()

    # Calculate SSIM
    score, _ = ssim(image1_gray_np, image2_gray_np, full=True, data_range=image2_gray_np.max() - image2_gray_np.min())
    return score
```

Prompt 2:
Are the uploaded images effectively show data augmentation and are neatly presented?

Response 2 ***(GPT 4o)*****:**

Data Augmentation

1. **Diversity of Augmentation:**
   - The augmented images show noticeable changes in orientation and possibly other transformations like zoom, shift, and shear. This suggests that multiple augmentation techniques were applied.
   - **Assessment:** The augmentation appears to be diverse and effective, making the images visibly different from the originals, which is crucial for enhancing model robustness.
2. **Quality and Realism:**
   - The augmented images maintain the visual quality and realism, which is important for training purposes.
   - **Assessment:** The images look realistic, and the transformations do not distort the objects in a way that would make them unusable for training.

Presentation

1. **Neatness:**
   - The images are aligned in a row, labeled clearly with titles indicating the originals and augmented versions.
   - **Assessment:** The presentation is neat and clear, making it easy to compare the original images with their augmented versions.
2. **Clarity:**
   - Each augmented image is labeled sequentially, which helps in identifying the different augmentation outputs.
   - **Assessment:** The clarity of labeling and layout is excellent, aiding in quick visual comparison.

NB: ***All data augmented images were uploaded to ChatGPT***

Prompt 3:
Provide insights on the box plot, violin plot, and histogram based on the three uploaded images.

Response 3 ***(GPT 4o)*****:**

Box Plot

- **Median and Quartiles:** The box plot shows that the median SSIM score of the augmented images is slightly above 0.8. The interquartile range (IQR), which represents the middle 50% of the data, ranges from approximately 0.75 to 0.82.

- **Outliers:** There are a few outliers in the data, with some SSIM scores as low as 0.5 and 0.6, indicating that some augmented images are significantly different from the original images.

Violin Plot

- **Distribution Shape:** The violin plot provides a detailed view of the SSIM score distribution. The density is highest around the median (0.8), suggesting that most augmented images have a high similarity to the original images.

- **Variability:** There is a wider spread towards the lower SSIM scores, indicating a subset of augmented images that are more significantly different from the originals. This tail suggests variability in the augmentation process.

Histogram

- **Frequency Distribution:** The histogram shows the frequency distribution of the SSIM scores. Most augmented images have SSIM scores between 0.7 and 0.85, with the highest frequency occurring around 0.8.

- **Bimodal Distribution:** There appears to be a bimodal distribution, with one peak around 0.8 and a smaller peak around 0.6, indicating two clusters of similarity levels among the augmented images.

The majority of the augmented images have high SSIM scores, suggesting successful augmentation with most images retaining significant similarity to the originals. The presence of outliers and a bimodal distribution indicates that some augmentations result in images that are more distinct from the originals.

NB: ***The box plot, violin plot and histrogram were all uploaded to ChatGPT***

Prompt 4:
Based on the uploaded Jupyter notebook, please assess whether it meets the bellow specified criteria? Guidelines: *'given marking scheme'*.

Response 4 *(GPT 4o)*:
Implementation of Data Augmentation in TensorFlow (15 marks)

1. Use of ImageDataGenerator class and transforms module to implement at least three image transformations (5 marks):
    - The notebook uses the ImageDataGenerator class from TensorFlow and implements various transformations: rotation, width shift, height shift, shear, zoom, and horizontal flip. **Score: 5/5**
2. Generation of at least 5 augmented images per input image (4 marks):
    - The code generates 5 augmented images for each input image using a loop. **Score: 4/4**
3. Proper handling of image data, including loading, processing, and saving to file (3 marks):
    - The notebook includes code to load images, process them through the augmentation pipeline, and save the augmented images to a specified directory. **Score: 3/3**
4. Efficient use of memory and computational resources (3 marks):
    - The use of ImageDataGenerator is memory-efficient, and there is no indication of inefficient memory or computational resource usage in the provided code. **Score: 3/3**

Quality of Augmented Images (5 marks)

1. Images are visibly different from the original images, indicating successful augmentation (3 marks):
    - The variety of transformations (rotation, shift, shear, zoom, flip) suggests that the augmented images will be visibly different from the original ones. **Score: 3/3**
2. Augmented images are in a compatible format and named in a way that is easy to identify (2 marks):
    - The augmented images are saved in JPEG format with names that include the original filename and an augmentation index. **Score: 2/2**

Total Score: 20/20

The notebook meets all the criteria outlined, with clear implementation and handling of data augmentation in TensorFlow, and the quality of augmented images being high. Therefore, it scores a perfect 20 marks.

NB: ***The Jupyter Notebook was fully uploaded to ChatGPT***

Prompt 5:
Please review the below text for grammar and punctuation errors, suggest rephrasing sentences or paragraphs for improved clarity and coherence, and recommend better word choices.
Text: '*Introduction of GenAI Journal*'

Response 5 *(Gemini 1.5 Pro)*:
The text is well-written overall, but here are some suggestions for improvement:

- **Conciseness:** The first sentence can be shortened.
  - **Original:** In the second section of our computer vision (CV) assignment, we investigate the dramatic impact of generative AI models on productivity and accuracy.
  - **Revised:** The second section of our computer vision assignment explores how generative AI models boost productivity and accuracy.
- **Specificity:** Consider mentioning the specific task(s) you performed in the assignment.
  - **Original:** We utilized various generative AI models, carefully selecting each to maximize their unique capabilities.
  - **Revised:** To explore code optimization and image analysis, we carefully selected various generative AI models to leverage their unique strengths.
- **Combining Paragraphs:** The paragraphs about GPT-4 and ChatGPT can be combined for better flow.
- **Word Choice:** "Outstanding" is used repeatedly. Consider synonyms like "impressive" or "effective."

NB: *The Introduction paragraph (from this GenAI Journal) was uploaded to Gemini*

4.2 Individual Reflection

Incorporating Generative AI into the project has greatly improved my technical and analytical skills. The process began with GenAI providing detailed feedback on data augmentation techniques, which helped me refine my approach. This included specific instructions for using TensorFlow's ImageDataGenerator effectively, ensuring that augmented images were both high-quality and accurate. The GenAI also conducted a thorough evaluation using the provided marking scheme, identifying areas for improvement with clarity and precision.

One of the most useful features was the GenAI's ability to generate in-line comments within my code, which improved the readability and clarity of my work while also ensuring that my coding practices met the highest standards. Furthermore, the AI meticulously reviewed my grammar, ensuring that my documentation and code comments were error-free, which improved overall communication and quality of my work.

Moreover, GenAI's extensive knowledge of visual data representations such as box plots, violin plots, and histograms greatly improved my understanding of data distribution and variability. The AI's insights into these visual tools enabled me to better interpret and present my data, resulting in more informed and precise analyses. In summary, incorporating generative AI into the project provided a diverse learning opportunity that enhanced both my technical execution and analytical depth.

---

5 Section: Thomas Koppens

5.1 Prompts & Responses

Prompt 1:
Tell me about the parameters I can change in pytorch's '*ColorJitter*' transform.

Response 1 *(GPT 4o)*:
In PyTorch, the ColorJitter transform is used for random changes in the brightness, contrast, saturation, and hue of an image. It is often used for data augmentation to make the model more robust to changes in lighting conditions. Here are the parameters you can change in the ColorJitter transform:

1. **Brightness**: Controls the brightness adjustment. This can be a single float value or a tuple of two float values (min, max). If it's a single value, the range will be (1 - value, 1 + value). If it's a tuple, the range will be exactly as specified.
2. **Contrast**: Controls the contrast adjustment. Like brightness, this can be a single float value or a tuple of two float values. The range follows the same logic as the brightness parameter.
3. **Saturation**: Controls the saturation adjustment. This can also be a single float value or a tuple of two float values, with the same range logic as the brightness and contrast parameters.
4. **Hue**: Controls the hue adjustment. This parameter should be in the range [-0.5, 0.5] for a single float value or a tuple of two float values within the same range. If it's a single float, the range will be (-value, +value).

Here's an example of how to use the ColorJitter transform with these parameters:

```
color_jitter = transforms.ColorJitter(brightness=0.2, # Brightness factor between 0.8 and 1.2
                                      contrast=0.3, # Contrast factor between 0.7 and 1.3
                                      saturation=0.4, # Saturation factor between 0.6 and 1.4
                                      hue=0.1 )# Hue factor between -0.1 and 0.1
```

You can also use tuples to provide specific ranges for each parameter:

```
color_jitter = transforms.ColorJitter(brightness=(0.8, 1.2), # Specific range for brightness
                                      contrast=(0.7, 1.3), # Specific range for contrast
                                      saturation=(0.6, 1.4), # Specific range for saturation
                                      hue=(-0.1, 0.1)) # Specific range for hue
```

These parameters allow you to adjust how much you want to vary the brightness, contrast, saturation, and hue of your images for data augmentation purposes.

Prompt 2:
Give me an example of how I can compare two images using ‘*ignite’s SSIM*‘.

Response 2 ***(GPT 4o)*****:**
To compare two images using the Structural Similarity Index (SSIM) with PyTorch-Ignite, you can use the `ignite.metrics.SSIM` class. SSIM is a perceptual metric that measures the similarity between two images.

Here is a step-by-step example of how to do this:

1. **Install PyTorch-Ignite:** If you haven’t already installed PyTorch-Ignite, you can do so using: **pip install pytorch-ignite**
2. **Load the images:** Load the images you want to compare.
3. **Preprocess the images:** Ensure that both images are in the same format (e.g., size, color channels) and converted to PyTorch tensors.
4. **Compute SSIM:** Use the `ignite.metrics.SSIM` to compute the similarity.

Here is a complete example:

```
import torch
import torchvision.transforms as transforms
from PIL import Image
from ignite.metrics import SSIM

# Load the images
image1 = Image.open(’image1.jpg’)
image2 = Image.open(’image2.jpg’)

# Define the transformations
transform = transforms.Compose([
    transforms.Resize((256, 256)),  # Ensure both images are the same size
    transforms.ToTensor()  # Convert images to PyTorch tensors
])

# Apply the transformations
image1_tensor = transform(image1).unsqueeze(0)  # Add batch dimension
image2_tensor = transform(image2).unsqueeze(0)  # Add batch dimension

# Ensure images are on the same device
device = torch.device(’cuda’ if torch.cuda.is_available() else ’cpu’)
image1_tensor = image1_tensor.to(device)
image2_tensor = image2_tensor.to(device)

# Initialize SSIM metric
ssim_metric = SSIM(data_range=1.0, device=device)  # data_range should match the range of your image tensors
# Update the metric with the images
ssim_metric.update((image1_tensor, image2_tensor))

# Compute SSIM
ssim_value = ssim_metric.compute()
print(f’SSIM between the two images: {ssim_value:.4f}’)
```

Explanation:

1. **Load the Images:** `Image.open` is used to load the images from disk.
2. **Transformations:** `transforms.Resize` ensures both images are the same size, and `transforms.ToTensor` converts the images to PyTorch tensors.
3. **Add Batch Dimension:** `unsqueeze(0)` adds a batch dimension to the tensors, as SSIM expects batched inputs.
4. **Move to Device:** Ensures that the images are on the same device (CPU or GPU).
5. **Initialize SSIM:** The `SSIM` class from `ignite.metrics` is initialized with the appropriate data range and device.
6. **Update and Compute SSIM:** The `update` method is called with the two images, and `compute` calculates the SSIM value.

Prompt 3:
What would be considered a good SSIM value where the image is different enough to improve robustness when training models, but close enough such that semantics are retained?

Response 2 ***(GPT 4o)*****:**
In the context of training models where robustness is a concern, a "good" SSIM value for augmented images would typically be one that indicates enough diversity to enhance model generalization while still preserving semantic information.

This can be somewhat subjective and application-dependent, but generally speaking, SSIM values in the range of 0.8 to 0.9 might be considered suitable. These values suggest sufficient diversity to improve model robustness by exposing it to a variety of inputs, while still ensuring that the fundamental semantic information is retained.

However, it's essential to consider the specific requirements of your application and the characteristics of your dataset. In some cases, higher SSIM values might be preferred if preserving fine-grained details is critical, while in other cases, lower SSIM values might be acceptable if the goal is to introduce more significant variations for robustness testing.

Ultimately, the choice of SSIM threshold should be guided by a balance between the need for diversity and the importance of preserving key semantic features in the training data. Experimentation and validation on your specific dataset and model can help determine the optimal SSIM threshold for your particular use case.

5.2 Individual Reflection

I have found that generative AI, namely ChatGPT, has been a very useful academic tool for me since I started using it seriously around my first year in the course. I realized its potential disadvantages for my experience at university very quickly. I had two main concerns:

1. Its effect on my mental process when tackling problems (since it would be easy to just turn to generative AI as soon as I faced any friction).
2. How it would impact how much I actually learned, since it could instill a habit where I would neglect learning things since it would be able to handle it for me.

Thus, I have always been wary about its usage. I always try to use it minimally and will usually turn to internet sources such as documentation or Stack Overflow threads first. However, I certainly still use it. In this project, I struggled to understand some PyTorch documentation, as well as finding solutions to how to save/load images in PyTorch. Generative AI successfully aided me by writing accurate, minimal working examples to solve such problems. Furthermore, it explained how certain parameters worked so that I could more effectively understand and use the transforms in PyTorch.

Interestingly, the explicit permission to use generative AI served as a challenge to use it less in my mind. I found that intentionally avoiding it during times when I was completely stuck only slowed down the process of working on this project. Eventually, I did indeed turn to it, and it helped. So, if I learned anything about gen AI during this project, it would be since it is a useful tool and source like any other, there is a time and a place to use it appropriately, and I should not be afraid of that.

---

6 Section: Sean David Muscat

6.1 Prompts & Responses

Prompt 1:

In the bellow code snippet, add further in-line comments:

```
os.makedirs(os.path.join(results_folder, ’Brightness’), exist_ok=True)

def adjust_brightness(image, brightness):
    bright_img = np.clip(image + brightness, 0, 255).astype(np.uint8)
    return bright_img

brightness_levels = [-100, -50, 0, 50, 100]

# Generate and display augmented images
for image_file, img in images:
    fig, axes = plt.subplots(1, len(brightness_levels) + 1, figsize=(20, 5))
    fig.suptitle(f’Brightness Variation for {image_file}’)

    # Display original image
    axes[0].imshow(img)
    axes[0].set_title(’Original’)
    axes[0].axis(’off’)

    for ax, level in zip(axes[1:], brightness_levels):
        augmented_img = adjust_brightness(img, level)
        ax.imshow(augmented_img)
        ax.set_title(f’Brightness {level}’)
        ax.axis(’off’)
        output_filename = os.path.join(results_folder, ’Brightness’, f’brightness_{level}_{image_file}’)
        plt.imsave(output_filename, augmented_img)

    plt.show()
```

Response 1 *(GPT 4o)*:

Added further inline comments for code clarity:

```
# Ensure the ’Brightness’ directory exists in the ’Results’ folder
os.makedirs(os.path.join(results_folder, ’Brightness’), exist_ok=True)

# Function to adjust brightness
def adjust_brightness(image, brightness):
    # Clip the values to be in the valid range [0, 255]
    bright_img = np.clip(image + brightness, 0, 255).astype(np.uint8)
    return bright_img

# Brightness levels to apply
brightness_levels = [-100, -50, 0, 50, 100]

# Generate and display augmented images
for image_file, img in images:
    fig, axes = plt.subplots(1, len(brightness_levels) + 1, figsize=(20, 5))
    fig.suptitle(f’Brightness Variation for {image_file}’)

    # Display original image
    axes[0].imshow(img)
    axes[0].set_title(’Original’)
    axes[0].axis(’off’)

    # Generate augmented images
    for ax, level in zip(axes[1:], brightness_levels):
        augmented_img = adjust_brightness(img, level)
        ax.imshow(augmented_img)
        ax.set_title(f’Brightness {level}’)
        ax.axis(’off’)

        # Save the augmented image in the ’Brightness’ directory within the ’Results’ folder
        output_filename = os.path.join(results_folder, ’Brightness’, f’brightness_{level}_{image_file}’)
        plt.imsave(output_filename, augmented_img)

    plt.show()
```

Prompt 2:
I have written the following paragraph to explain the above code. Can you check it for me, making sure it is factual and grammatically correct?

"The code starts by ensuring that augmented photographs will be stored in a directory called "contrast." Then, it defines the adjust_ contrast() function to alter the contrast levels of images. This function accepts an image and a contrast parameter as inputs. It calculates the midpoint value (which is set to 128) and applies a contrast adjustment formula to each pixel. The formula enhances or reduces contrast based on the given contrast parameter. The modified image is then returned by the function. Following this, a list of contrast levels ranging from 0.5 to 1.5 is established. These levels represent the variations in contrast that will be applied to the photos.

Lastly, a set of augmented images is generated for each image by adjusting the contrast levels according to the specified list. These enhanced images are displayed alongside the original image, allowing for visual comparison of different contrast levels. The higher the contrast value given, the more "distorted" the image appears and the more certain colours (like the large pink found for the volcano object) pop out."

Response 2 ***(Gemini 1.5 Pro)*****:**
Your paragraph is clear and informative, but we can refine it for better flow and clarity. Here's a revised version:

The script begins by ensuring the existence of a directory named "brightness" where the augmented photographs will be stored. Following this, it defines the adjust_ brightness() function, responsible for altering the brightness levels of images. This function takes an image and a brightness parameter as inputs, adjusts each pixel's value by the given parameter to modify brightness, and ensures that pixel values remain within the acceptable range of 0 to 255. Subsequently, a list encompassing brightness levels ranging from -100 to 100 is established to represent the variations that will be applied to the photos.

Once the list is set, a series of augmented images is generated for each original image, with their brightness levels adjusted according to the specified list. These modified images are juxtaposed with the originals to facilitate visual comparison across various brightness levels. It's important to note that alterations to brightness levels significantly impact the visual appearance of photographs. Positive values increase brightness, negative values decrease it, while a value of 0 leaves the image unchanged. The magnitude of the value dictates the extent of the change, with more noticeable alterations occurring at higher values and subtler changes at lower ones. Experimenting with different values allows for precise adjustment of brightness to achieve desired visual effects.

NB: ***I ended up changing my paragraph after taking Gemini's consideration on board.***

Prompt 3:
I've used your paragraph and have even edited it. Can you now act as a peer-reviewer and review it for me. *'Revised paragraph here'*

Response 3 ***(Gemini 1.5 Pro)*****:**
Your revised paragraph effectively communicates the process of adjusting brightness levels in images and provides clear instructions for the augmentation procedure. Here are some specific feedback points:

1. **Clarity:** The paragraph is well-structured and easy to follow, with each step of the process clearly explained. Readers can easily understand the purpose of the script and how it operates.
2. **Conciseness:** The language used is concise and avoids unnecessary repetition or verbosity. Each sentence contributes to conveying important information without overwhelming the reader with unnecessary details.
3. **Technical Accuracy:** The paragraph accurately describes the function of the `adjust_brightness()` function and how it modifies pixel values within the acceptable range. The explanation of how positive and negative brightness values affect image appearance is clear and accurate.
4. **Flow:** The flow of the paragraph is smooth, leading the reader logically through each step of the process. The transition between explaining the function and establishing the list of brightness levels is seamless.
5. **Engagement:** While the paragraph effectively conveys technical information, it could be enhanced with a touch of engagement to make it more engaging for readers. Consider adding a sentence or phrase that highlights the practical application or significance of adjusting brightness levels in image processing.

Overall, your revised paragraph effectively communicates the process well. With a minor addition to enhance engagement, it would be even more compelling for readers.

Prompt 4:
I have generated the following imagesfor the distortion Shearing. Can you verify if they are correct?

Response 4 ***(GPT 4o)*****:**
Yes, the sheared images appear to be generated correctly according to the specified shear factors. Here's a detailed verification based on your provided image:

1. **Shear (0.2, 0.0):** This should shear the image in the x-direction. The top rows of the image should move to the right more than the bottom rows, creating a rightward slant. Your images in the first row show this effect.
2. **Shear (0.0, 0.2):** This should shear the image in the y-direction. The right columns of the image should move down more than the left columns, creating a downward slant. Your images in the second row show this effect.
3. **Shear (0.2, 0.2):** This should shear the image in both x and y directions, creating a diagonal distortion. Your images in the third row show this combined effect.
4. **Shear (-0.2, 0.0):** This should shear the image in the negative x-direction. The top rows of the image should move to the left more than the bottom rows, creating a leftward slant. Your images in the fourth row show this effect.
5. **Shear (0.0, -0.2):** This should shear the image in the negative y-direction. The right columns of the image should move up more than the left columns, creating an upward slant. Your images in the fifth row show this effect.

The images in your output demonstrate these shearing transformations clearly, suggesting that the shearing function is working correctly.

NB 1: ***The images were uploaded to ChatGPT***
NB 2: ***This query was asked for all data augmentation techniques***

Prompt 5:
My previous file structure has changed (where I load my images from). Please provide a template for folder navigation; to read and save images more dynamically?

Response 5 ***(GPT 4.0)*****:**

Certainly! It’s a good practice to make your file paths more dynamic and adaptable, especially if your file structure may change in the future. Here’s a template that you can use as a guideline:

```
# Ensure the ’Brightness’ directory exists in the ’Results’ folder
os.makedirs(os.path.join(results_folder, ’Brightness’), exist_ok=True)
```

```
# Save the augmented image in the ’Brightness’ directory within the ’Results’ folder
output_filename = os.path.join(results_folder, ’Brightness’, f’brightness_{level}_{image_file}’)
plt.imsave(output_filename, augmented_img)
```

6.2 Individual Reflection

My section was Q4 and overall, generative AI was mainly used to check the output pictures I had gotten from my functions. After that, it was used sparingly, mainly to help with writing or for general code refactoring to make it more efficient. It helped cover some of my weaknesses when it came to concise and “tidy” code that is easily readable by others, so using generative AI helped keep me on track there and verify that everything made sense. So overall, generative AI helped strengthen my pre-established foundations throughout the notebook and made it a more efficient process throughout.

7 Improvements and Contributions

Generative AI significantly improved many aspects of our work, particularly the quality of our code and the clarity of our inline comments. These enhancements not only improved the quality of our technical output, but also helped to address the ethical concerns outlined in our journal.

In tasks that required image outputs, generative AI was a valuable addition to our data analysis, acting as a secondary validation mechanism to ensure the accuracy of our results. It also played an important role in improving the quality of our written content by acting as a peer reviewer, ensuring that our ideas were effectively articulated in written form. Furthermore, when faced with challenges during the project, the AI proved to be an invaluable resource, providing alternative perspectives and assisting with problem-solving efforts.

Overall, generative AI served as a versatile tool, supplementing our work in a variety of domains such as data analysis, ethical considerations, and content creation, thereby improving the overall quality and efficacy of our project outcomes.

8 References

[1] OpenAI, "ChatGPT," ChatGPT. [Online]. Available: https://chatgpt.com/?oai-dm=1. [Accessed: 06-Jun-2024].

[2] OpenAI, "ChatGPT," ChatGPT. [Online]. Available: https://openai.com/chatgpt/. [Accessed: 06-Jun-2024].

[3] OpenAI, "ChatGPT," ChatGPT. [Online]. Available: https://openai.com/index/chatgpt/. [Accessed: Jun. 6, 2024].

[4] Google, "Google Gemini," Google Gemini. [Online]. Available: https://gemini.google.com/app. [Accessed: Jun. 6, 2024].

[5] DeepMind, "Gemini," DeepMind. [Online].
Available: https://deepmind.google/technologies/gemini/. [Accessed: Jun. 6, 2024].

[6] Google, "Google Gemini AI," Google Blog. [Online].
Available: https://blog.google/technology/ai/google-gemini-ai/. [Accessed: Jun. 6, 2024].

[7] Amazon Web Services, "What is Data Augmentation?," AWS. [Online].
Available: https://aws.amazon.com/what-is/data-augmentation/. [Accessed: Jun. 6, 2024].

[8] S. Das, "Ethics in Generative AI: Concerns, Examples, and Best Practices," Ellow, Apr. 1, 2024. [Online]. Available: https://ellow.io/ethics-in-generative-ai/. [Accessed: Jun. 6, 2024].

[9] M. Zhou, et al., "Bias in Generative AI," arXiv, General Economics (econ.GN), Mar. 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2403.02726. [Accessed: Jun. 6, 2024].

[10] V. Kothandapani, "Addressing bias in generative AI starts with training data explainability," RWS, Mar. 8, 2024. [Online]. Available: https://www.rws.com/artificial-intelligence/train-ai-data-services/blog/address-bias-with-generative-ai-data-explainability/. [Accessed: Jun. 6, 2024].

[11] L. R. Salazar, et al., "Generative AI Ethical Considerations and Discriminatory Biases on Diverse Students Within the Classroom," in The Role of Generative AI in the Communication Classroom, pp. 191-213, Feb. 2024, doi: 10.4018/979-8-3693-0831-8.ch010.

[12] Z. Argento, "Data protection issues for employers to consider when using generative AI," IAPP, Aug. 9, 2023. [Online]. Available: https://iapp.org/news/a/data-protection-issues-for-employers-to-consider-when-using-generative-ai. [Accessed: Jun. 6, 2024].

[13] A. Golda, K. Mekonen, A. Pandey, and A. Singh, "Privacy and Security Concerns in Generative AI: A Comprehensive Survey," IEEE Access, vol. PP, no. 99, pp. 1-1, Jan. 2024, doi: 10.1109/ACCESS.2024.3381611.

[14] CEDPO AI Working Group, "Generative AI: The Data Protection Implications," Confederation of European Data Protection Organisations (CEDPO), Oct. 16, 2023. [Online]. Available: https://cedpo.eu/wp-content/uploads/generative-ai-the-data-protection-implications-16-10-2023.pdf. [Accessed: Jun. 6, 2024].

[15] J. Zhou, et al., "Synthetic Lies: Understanding AI-Generated Misinformation and Evaluating Algorithmic and Human Solutions," in CHI '23: Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, Hamburg, Germany, Apr. 2023, doi: 10.1145/3544548.3581318.

8.1 COTS Dataset

[16] D. Seychell, C. J. Debono, M. Bugeja, J. Borg, and M. Sacco, "COTS: A Multipurpose RGB-D Dataset for Saliency and Image Manipulation Applications," IEEE Access, vol. 9, pp. 21481-21497, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9340352. [Accessed: Jun. 6, 2024].

[17] D. Seychell, C. J. Debono, M. Bugeja, J. Borg, and M. Sacco, "COTS: A Multipurpose RGB-D Dataset for Saliency and Image Manipulation Applications," GitHub, 2021. [Online]. Available: https://github.com/dylanseychell/COTSDataset. [Accessed: Jun. 6, 2024].
