Matthias Vassallo Pulis, Jacob Zammit

Generative AI Journal

Introduction

Throughout the project, only ChatGPT was used as the generative AI model. There were many reasons behind this choice. One of the main reasons is because it is the most popular out of all the existing AI models. Another major reason is that its performance has improved in recent times, particularly with the release of a new model, GPT-4o, which greatly helped in Part B of the project [1]. Apart from that, we utilised the above-mentioned generative AI model for its advanced language processing capabilities, ease of use, and for its ability to enhance the quality of our work by providing insights through the responses it gives. Overall, this AI model was chosen in order to improve the quality of our project.

Ethical Considerations

Before using ChatGPT, we made sure that we understood the ethical aspects of using generative AI. This includes data privacy and bias. Privacy was maintained by checking every prompt that was entered to ensure that there was no sensitive data inputted. Data bias was reduced by carefully analysing the responses generated by ChatGPT to ensure that there was no bias, which would affect our project's findings. We also ensured that the content generated by the AI model was used to guide us with our work, rather than replace it, thereby sustaining the integrity and authenticity of our project.

Methodology

Several steps were taken to integrate generative AI into our project in a fair way. We had two approaches for this, in order to meet the standards required for each task.

In Part A, we first wrote the background about data augmentation without using generative AI, as stated in Task 1 of the project. Afterwards, once the whole background has been written, we copied the entire writing and pasted it onto the ChatGPT prompt textbox. Within the same textbox, we also made sure to state that we wanted to ask the generative AI model several questions regarding our writing. Then, we asked the AI model a simple question on how it can be improved, for example on grammar or punctuation. We repeated the same process for every new prompt entered. Once the responses for each question have been generated, we made sure to analyse them carefully to ensure that they are deemed very helpful for improving our writing. We also made sure to document each prompt and its response appropriately.

Our approach for Part B was more structured as this is divided into different steps. Firstly, we made sure that we had access to ChatGPT, which, as stated above, was our AI model of choice. Then, if we got stuck on something or if we wished to understand something in more detail, we gave it a prompt. In the prompt, we made sure to emphasise the type of response that would be given in order to maximise the accuracy of its responses. Afterwards, once the response has been generated, we carefully read it to ensure that it is aligned with our needs according to our project. Finally, we applied them in an appropriate manner, making sure to document each prompt and response that has guided us throughout the development of both tasks in this part.

Prompts and Responses

Part A

Prompt 1:
Are there any suggestions on improving the grammar of this writing?

Reply 1:
Sure, here are some suggestions to improve the grammar and clarity of the writing:

1. Introduction:
    - Consider rephrasing "Data augmentation is defined as artificially increasing the size of training datasets by making modified copies of a dataset using existing data" to "Data augmentation is the process of artificially increasing the size of training datasets by creating modified copies of existing data."
    - Instead of "The way this augmented data is created is by making certain minor changes to the dataset", try "This augmented data is created by making specific minor changes to the dataset."
2. Historical Context:
    - Instead of "Then a few years later, in the decades of the 1980s and the 1990s", consider "Subsequently, during the 1980s and 1990s".
    - Instead of "then popularising the field of computer vision", try "which popularized the field of computer vision".
3. Data Augmentation Techniques:
    - Consider adding a comma after "Geometric Transformations" for clarity.

   - Instead of "colour Space Transformations", consider "Color Space Transformations" (assuming American English spelling).
   - Instead of "This technique is called ‘Salt and Pepper Noise’", consider "This technique is called 'Salt and Pepper Noise'."
4. Recent Advancements:
   - Instead of "This is used in data augmentation to increase the amount of labelled data in graphs, which in turn makes the size of the training dataset grow", try "This approach is utilized in data augmentation to increase the quantity of labeled data in graphs, consequently expanding the size of the training dataset."
5. Applications in Computer Vision:
   - Consider rephrasing "making use of basic augmentation techniques, such as image cropping, rotation and adjusting brightness" to "utilizing fundamental augmentation techniques, such as image cropping, rotation, and brightness adjustment".
   - Instead of "Basic techniques are again put in place, especially flipping and adjusting the intensity of the image", try "Fundamental techniques are employed, particularly flipping and adjusting the image's intensity."

These suggestions aim to enhance clarity and readability by refining sentence structure and word choice.

Prompt 2:
Would you also suggest any corrections to the punctuation?

Reply 2:
Certainly! Here are some suggestions for punctuation corrections:
1. Introduction:
   - Add a comma after "Data augmentation is defined as artificially increasing the size of training datasets by making modified copies of a dataset using existing data".
2. Historical Context:
   - Instead of "Subsequently, during the 1980s and 1990s", try "Subsequently, during the 1980s and 1990s,".
3. Data Augmentation Techniques:
   - Consider adding a comma after "Geometric Transformations" for clarity.
   - Instead of "colour Space Transformations", consider "Color Space Transformations,".
4. Recent Advancements:

  - Instead of "This approach is utilized in data augmentation to increase the quantity of labeled data in graphs, consequently expanding the size of the training dataset", try "This approach is utilized in data augmentation to increase the quantity of labeled data in graphs, consequently expanding the size of the training dataset.".
5. Applications in Computer Vision:
    - Consider adding a comma after "utilizing fundamental augmentation techniques, such as image cropping, rotation, and brightness adjustment".

These punctuation suggestions aim to ensure proper grammatical structure and aid in readability by clarifying the flow of ideas within the sentences.

Prompt 3: Please give me some suggestions on improving the writing, such as grammar and punctuation corrections, rephrasing sentences or paragraphs, and suggestions for better word choices.

Output 3:
Certainly! Here are some suggestions to enhance the writing in your essay:

Grammar and Punctuation Corrections:
Ensure consistency in punctuation usage throughout the essay, such as commas, periods, and semicolons.
Double-check subject-verb agreement and eliminate any grammatical errors.
Rephrasing Sentences or Paragraphs:
Consider restructuring complex sentences into shorter, more concise ones to improve readability.
Clarify any ambiguous phrasing or unclear expressions to enhance comprehension.
Better Word Choices:
Opt for precise and descriptive language to convey ideas effectively.
Use synonyms or alternate phrases to avoid repetition and add variety to your writing.

Prompt 4:
Are there any sentences that need rephrasing to make the reader understand the context better?

Reply 4:

Certainly! Here are some sentences that could be rephrased to enhance clarity and improve the reader's understanding:

1. Introduction:
    - Original: "Data augmentation is defined as artificially increasing the size of training datasets by making modified copies of a dataset using existing data."
    - Rephrased: "Data augmentation involves artificially expanding the size of training datasets by creating modified copies of existing data."
2. Data Augmentation Techniques:
    - Original: "Geometric Transformations are modifications to the spatial properties of images in order to create different perspectives, orientations, and scales."
    - Rephrased: "Geometric Transformations modify the spatial properties of images to generate different perspectives, orientations, and scales."
3. Recent Advancements:
    - Original: "Approaches involving Graph-Neural Networks (GNNs) have emerged as a popular strategy for handling graph data due to their ability to include structural context."
    - Rephrased: "Graph-Neural Networks (GNNs) have become a popular strategy for handling graph data due to their ability to incorporate structural context."
4. Applications in Computer Vision:
    - Original: "Image segmentation is another Computer Vision application which uses data augmentation."
    - Rephrased: "Image segmentation, another application in Computer Vision, utilizes data augmentation."

These rephrased sentences aim to convey the same meaning but in a clearer and more concise manner, making it easier for the reader to understand the context.

Prompt 5:
Give me feedback on the overall structure and flow of the writing, as well as suggestions for any additional content that could be added to improve the quality of my writing.

Output 5:
Overall, the structure of your essay is well-organized, with clear sections addressing different aspects of data augmentation in computer vision. The introduction effectively sets the stage for the topic, and each subsequent section delves into

relevant details, such as historical context, data augmentation techniques, recent advancements, and applications in computer vision.

However, to further improve the flow of your writing, consider refining the transitions between paragraphs to ensure a smoother transition between ideas. Additionally, you might want to provide more detailed examples or case studies to illustrate key concepts and enhance reader understanding.

In terms of additional content, you could consider discussing potential challenges or limitations associated with data augmentation in computer vision, as well as emerging trends or future directions in the field. This would provide readers with a more comprehensive overview and stimulate further discussion on the topic.

Prompt 6: Now act as a strict peer-reviewer and give me the pros and cons of your work.

Output 6: As a strict peer-reviewer, here are the pros and cons of your work:

Pros:

Comprehensive Coverage: Your essay provides a thorough overview of data augmentation in computer vision, covering its definition, historical context, techniques, recent advancements, and applications.
Well-Structured: The essay is organized into clear sections, making it easy for readers to follow the flow of ideas and locate relevant information.
Evidence-Based: You support your arguments with references to credible sources, adding credibility to your analysis and insights.
Clarity: The writing is clear and concise, with explanations presented in a straightforward manner, enhancing reader comprehension.
Cons:

Lack of Depth in Historical Context: While you briefly touch upon milestones in computer vision, such as the Canny Edge Detection algorithm and the Hough Transform technique, you could provide more in-depth analysis and discuss their specific contributions to the field.
Limited Discussion of Recent Advancements: Although you mention advancements involving Graph-Neural Networks (GNNs) and Convolutional Neural Networks (CNNs), you could delve deeper into how these technologies are specifically applied in data augmentation and their impact on computer vision.
Minimal Exploration of Challenges or Limitations: Your essay primarily focuses on the benefits and applications of data augmentation, but lacks discussion on potential challenges, limitations, or ethical considerations associated with its implementation.

Overall, your work demonstrates a strong understanding of the topic, but could benefit from deeper analysis and exploration of certain aspects to provide a more comprehensive overview.

Part B

Question 3:

Prompt 1: import os import numpy as np from tensorflow.keras.preprocessing.image import ImageDataGenerator, img_to_array, load_img import random # Define paths image_path = 'C:/Users/jacob/Desktop/COTSDATASET' tf_augmented_path = 'C:/Users/jacob/Desktop/CVPICS_TF' os.makedirs(tf_augmented_path, exist_ok=True) image_files = [f for f in os.listdir(image_path) if f.lower().endswith(('png', 'jpg', 'jpeg'))] random_images = random.sample(image_files, 5) print(f"Selected images: {random_images}") tf_datagen = ImageDataGenerator(rotation_range=40, width_shift_range=0.2, height_shift_range=0.2, shear_range=0.2, zoom_range=0.2, horizontal_flip=True, fill_mode='nearest') for image_file in random_images: img = load_img(os.path.join(image_path, image_file)) x = img_to_array(img) x = x.reshape((1,) + x.shape) i = 0 for batch in tf_datagen.flow(x, batch_size=1, save_to_dir=tf_augmented_path, save_prefix=f'tf_aug_{os.path.splitext(image_file)[0]}', save_format='jpeg'): i += 1 if i >= 3: break print("saved to", tf_augmented_path) is this code good?

Output 1:
The code you provided is a Python script for data augmentation using TensorFlow's Keras API. Let's review it:

Imports: You've imported necessary modules (os, numpy, ImageDataGenerator, img_to_array, load_img, random), which is good.

Paths and Directories: You've defined input and output directories for images. You've also ensured that the output directory exists or is created if it doesn't exist. This is good practice.

Image Selection: You're selecting random images from the input directory, which is fine if you want to apply augmentation to a subset of your dataset.

ImageDataGenerator: You've configured an ImageDataGenerator for image augmentation. The parameters you've chosen (rotation_range, width_shift_range, height_shift_range, shear_range, zoom_range, horizontal_flip, fill_mode) are commonly used for data augmentation and provide a good variety of transformations.

Augmentation Loop: You're looping through the selected images, applying augmentation, and saving the augmented images to the output directory. You've set a limit of 3 augmented images per original image, which seems reasonable.

File Extension Handling: You're using os.path.splitext to extract the filename and extension of the image file, which is good for ensuring that the augmented files have the correct format.

Print Statement: You're printing the path where the augmented images are saved, which is helpful for debugging.

Overall, the code looks good for its purpose of augmenting images using TensorFlow's Keras API. However, it's always a good idea to test the augmented images to ensure that the transformations are appropriate for your specific dataset and task.

Prompt 2:

import os import random from PIL import Image import torchvision.transforms as transforms image_path = 'C:/Users/jacob/Desktop/COTSDATASET' pytorch_augmented_path = 'C:/Users/jacob/Desktop/CVPICS_PYTORCH' os.makedirs(pytorch_augmented_path, exist_ok=True) image_files = [f for f in os.listdir(image_path) if f.lower().endswith(('png', 'jpg', 'jpeg'))] random_images = random.sample(image_files, 5) print(f"Selected images: {random_images}") pytorch_transform = transforms.Compose([transforms.RandomRotation(40), transforms.RandomHorizontalFlip(), transforms.RandomResizedCrop(256, scale=(0.8, 1.0)), transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.2)]) for image_file in random_images: img = Image.open(os.path.join(image_path, image_file)) for i in range(3): augmented_img = pytorch_transform(img) save_path = os.path.join(pytorch_augmented_path, f'pytorch_aug_{os.path.splitext(image_file)[0]}_{i}.jpeg') augmented_img.save(save_path) print("saved to", pytorch_augmented_path) check my code please

Output 2:

Your code seems fine for augmenting images using PyTorch's transforms module. Here's a quick rundown of what your code does:

Imports necessary libraries/modules (os, random, PIL, torchvision.transforms).
Defines paths for input images and output augmented images.
Retrieves image files from the specified directory.

Selects 5 random images from the list of image files.
Defines a sequence of transformations using torchvision.transforms.Compose(). These transformations include random rotation, horizontal flip, random resized crop, and color jitter.
Iterates over the randomly selected images, applies the defined transformations three times, and saves the augmented images with appropriate filenames in the specified output directory.
Here are some suggestions for improvement or things to consider:

Error Handling: Consider adding error handling mechanisms, especially when working with file operations (os.makedirs() or file I/O operations like opening and saving images).
Input Validation: Ensure that the input directory contains valid image files and handle cases where it might not.
Documentation: Adding comments or docstrings to explain the purpose of your code and each section of your code can improve its readability and maintainability.
Testing: Testing your code thoroughly with different inputs can help ensure its robustness.
Overall, your code looks good for its intended purpose of augmenting images using PyTorch's transforms.

Question 4:

Prompt 1: I just noticed, where there is mentioned, "Implement these techniques from first principles, hence not calling a function that simply carries out the distortion", does this mean I should not use the OpenCV library, also known as cv2 in Python?

Output 1: Yes, you're correct. The instruction "Implement these techniques from first principles, hence not calling a function that simply carries out the distortion" suggests that you shouldn't rely on pre-built functions or libraries like OpenCV's `cv2` module to directly apply the distortions.

Instead, you're expected to implement the distortion techniques manually, without relying on pre-existing functions that perform the distortion for you. This means you'll need to understand the underlying principles of each distortion technique and write the code to apply them from scratch.

For example, if you're implementing rotation, you wouldn't simply call a function like cv2.rotate(); instead, you'd write the code to manually rotate each pixel in the image using matrix transformations.

Similarly, for photometric distortions like brightness variation or contrast variation, you would calculate the new pixel values based on the desired distortion effect rather than using built-in functions.

This approach allows you to demonstrate a deeper understanding of the distortion techniques and gives you more control over the implementation process. It may require more coding effort, but it can also be a valuable learning experience.

Prompt 2: But this is what is mentioned: Using OpenCV, implement a program in a Jupyter Notebook that applies photometric and geometric distortions to the same subset of images used in Question 3. Apply at least three different techniques and three geometric distortions for photometric distortions. Implement these techniques from first principles, hence not calling a function that simply carries out the distortion.

Output 2: In Task 4, where you're asked to implement photometric and geometric distortions using OpenCV, you are indeed expected to use OpenCV for the implementation. However, the instruction to implement the techniques "from first principles" means that you should not simply rely on built-in functions that directly apply the distortions.

Instead, you should use OpenCV to perform the transformations but implement the logic for each distortion manually. This means understanding the underlying principles of each distortion technique and writing the code to apply them using basic image processing operations provided by OpenCV.

For example, if you're implementing rotation using OpenCV, you would not directly call cv2.rotate() but instead write the code to perform the rotation using matrix transformations or other relevant techniques.

Similarly, for photometric distortions like brightness variation or contrast variation, you would use OpenCV functions to manipulate pixel values but implement the logic for the distortion yourself rather than relying on a single function call that does everything for you.

So, in summary, you should use OpenCV for the implementation in Task 4, but make sure to implement the distortion techniques manually, without relying on functions that directly carry out the distortion with a single call.

Prompt 3: So what should I use instead to implement rotation for example?

Output 3: To implement rotation manually without relying on the cv2.rotate() function, you can use affine transformations. Here's a basic outline of how you can implement rotation using OpenCV:

1. **Calculate Rotation Matrix:** Determine the rotation angle (in degrees) and calculate the rotation matrix using the cv2.getRotationMatrix2D() function.
2. **Apply Affine Transformation:** Use the rotation matrix to perform the affine transformation on the image using the cv2.warpAffine() function.

Here's a Python code example demonstrating how to implement rotation manually [2]:

```
import cv2
import numpy as np


# Load the image
image = cv2.imread('input_image.jpg')


# Define rotation angle (in degrees)
angle = 30


# Calculate image center
height, width = image.shape[:2]
center = (width / 2, height / 2)


# Calculate rotation matrix
rotation_matrix = cv2.getRotationMatrix2D(center, angle, 1.0)


# Apply affine transformation to rotate the image
rotated_image = cv2.warpAffine(image, rotation_matrix, (width, height))
```

```
# Display the original and rotated images
cv2.imshow('Original Image', image)
cv2.imshow('Rotated Image', rotated_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In this example, angle specifies the rotation angle in degrees. The cv2.getRotationMatrix2D() function calculates the rotation matrix, and then the cv2.warpAffine() function applies the rotation to the image.

This approach allows you to implement rotation from scratch using OpenCV, fulfilling the requirement to implement the distortion technique manually.

Prompt 4: Now how can I also implement a way to change image brightness, saturation and contrast using similar functionality?

Output 4: You can adjust image brightness, saturation, and contrast using OpenCV by manipulating the pixel values directly. Here's how you can implement each of these adjustments manually:

1. **Image Brightness Adjustment:**
    - To adjust image brightness, you can simply add or subtract a constant value from each pixel intensity.
2. **Image Saturation Adjustment:**
    - To adjust image saturation, you can convert the image from BGR to HSV color space, then scale the saturation channel.
3. **Image Contrast Adjustment:**
    - To adjust image contrast, you can multiply each pixel intensity by a scaling factor.

Here's an example demonstrating each of these adjustments [3]:

```
import cv2
import numpy as np

# Load the image
```

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


# Image Brightness Adjustment
brightness_factor = 50  # Adjust brightness by adding a constant value
brightened_image = np.clip(image.astype(np.int16) + brightness_factor, 0,
255).astype(np.uint8)


# Image Saturation Adjustment
saturation_factor = 1.5  # Adjust saturation by scaling the saturation channel
hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
hsv_image[:, :, 1] = np.clip(hsv_image[:, :, 1] * saturation_factor, 0,
255).astype(np.uint8)
saturated_image = cv2.cvtColor(hsv_image, cv2.COLOR_HSV2BGR)


# Image Contrast Adjustment
contrast_factor = 1.5  # Adjust contrast by multiplying each pixel intensity by a
scaling factor
contrasted_image = np.clip(image.astype(np.float32) * contrast_factor, 0,
255).astype(np.uint8)


# Display the original and adjusted images
cv2.imshow('Original Image', image)
cv2.imshow('Brightened Image', brightened_image)
cv2.imshow('Saturated Image', saturated_image)
cv2.imshow('Contrasted Image', contrasted_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In this example, brightness adjustment is done by adding a constant value to each pixel intensity. Saturation adjustment involves scaling the saturation channel after converting the image to HSV color space. Contrast adjustment is achieved by multiplying each pixel intensity by a scaling factor. These manual adjustments allow you to control the brightness, saturation, and contrast of the image directly, fulfilling the requirement to implement the techniques from first principles.

Improvements and Contributions

The use of generative AI in our project significantly improved the quality of our project in several different areas. Firstly, it helped us improve the clarity of our background about data augmentation, which took place in Part A. This is mainly due to the valuable responses that were generated by ChatGPT, allowing us to improve the readability and clarity of our report.

When it comes to Part B, generative AI helped us to understand more about performing at least one distortion without always relying on using OpenCV’s built-in functions to perform it. This allowed us to experiment with specific distortions, especially the geometric ones, as they utilise a matrix to be able to perform the necessary transformations. It also helped in simplifying the code, especially when displaying each of the augmented images in the notebook.

Individual Reflection

Matthias’s Reflection

Upon using GenAI throughout this project, I found this tool extremely beneficial, especially in the way it offered valuable insights into improving grammar and structure, especially in Part A. This allowed the readability and clarity of the writing to improve. However, I encountered some problems with GenAI while working on the writing. For example, the writing of some English words were in the style of American English, and not British English.

In Part B, it introduced me to various ways of implementing the distortions mentioned in the project using first principles. In fact, I particularly learnt of two OpenCV functions to perform each of the geometric distortions implemented in the project, which are called cv2.warpAffine() and cv2.warpPerspective(). I also learnt of another function used to perform each of the photometric distortions implemented in the project, which is called cv2.convertScaleAbs(). The only problem I encountered while using GenAI for this part was that the parameter values for the cv2.convertScaleAbs() function would sometimes get switched. This made me get confused on the difference between brightness and contrast. I encountered the same problems as above but in a less impactful way, due to the upgraded GPT-4o model.

Overall, I learnt that while GenAI can be a great tool for helping one's writing, it should not be used too often, as it may not work as intended and it might limit one's creativity. Because of this reasoning, my perspective on using AI in academic projects has not changed much.

Jacob's Reflection

For Part A, I used ChatGPT as my choice of generative AI. One of the ways ChatGPT helped me was asking it to finish my sentence for me. This ensured I get my point across to the reader whenever I was struggling to find the exact words I wanted to say. I learned that it is possible to ask ChatGPT to rephrase a sentence or phrase. ChatGPT also helped by giving a skeleton of how the individual sections of the background should look like. At the end, I asked ChatGPT for feedback and it said the background was well-organized, but could have included additional details. This of course is unfeasible considering it needs to be a brief background with a 400 word limit, therefore I concluded my work was ready.

For Part B, I also chose to use ChatGPT, as it has proven time and time again to be the most effective and consistent Generative AI tool that is free to use. I used it to check the code I had written for the Tensorflow jupyter notebook and the Pytorch jupyter notebook. When I had done this in the past, it would usually bring up any obvious logical errors that my code contains, but for the two jupyter notebooks, it gave me the green flag, which is a good extra confirmation that the programs work correctly. I feel like this might become a step as important as running and testing your programs in the near future.

References and List of Resources Used

[1] OpenAI, “Introducing GPT-4o and more tools to ChatGPT free users,” OpenAI, https://openai.com/index/gpt-4o-and-more-tools-to-chatgpt-free (accessed Jun. 4, 2024).

[2] “Image Transformations using OpenCV in Python,” GeeksforGeeks, https://www.geeksforgeeks.org/image-transformations-using-opencv-in-python/ (accessed May 18, 2024).

[3] “Image Enhancement Techniques using OpenCV - Python,” GeeksforGeeks, https://www.geeksforgeeks.org/image-enhancement-techniques-using-opencv-python/ (accessed May 18, 2024).