

```

1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 # Function to resize images to a fixed size
6 def resize_image(img, target_size=(300, 300)):
7     return cv2.resize(img, target_size)
8
9 # Function to process images
10 def process_images(image_paths):
11     plt.figure(figsize=(20, 10))
12
13     for i, image_path in enumerate(image_paths, start=1):
14         # Read the image using OpenCV
15         img = cv2.imread(image_path)
16
17         if img is None:
18             print(f"Failed to load image {image_path}")
19             continue
20
21         # Resize image to a fixed size
22         resized_img = resize_image(img)
23
24         # Plot input image
25         plt.subplot(1, len(image_paths), i)
26         plt.imshow(cv2.cvtColor(resized_img, cv2.COLOR_BGR2RGB))
27         plt.title(f'Input Image {i}')
28         plt.axis('off')
29
30     plt.tight_layout()
31     plt.show()
32
33 # Example usage:
34 image_paths = ["/content/baby_image.jpeg", "/content/girlimage.jpeg", "/content/grandimage.jpeg", "/content/littleboy.jpg"]
35 process_images(image_paths)
36

```

Input Image 1



Input Image 2



Input Image 3

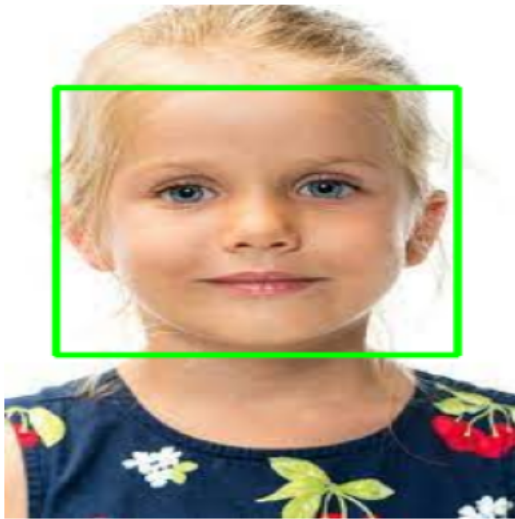


Input Image 4

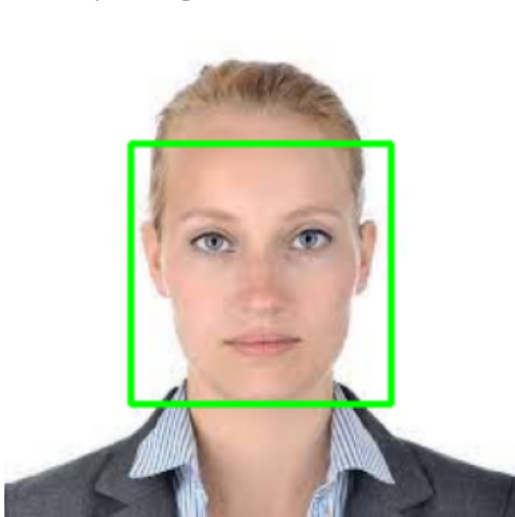


```
1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 # Function to detect faces and draw bounding boxes
6 def detect_faces(image_paths, target_size=(300, 300)):
7     for i, image_path in enumerate(image_paths, start=1):
8         # Read the image using OpenCV
9         img = cv2.imread(image_path)
10
11         if img is None:
12             print(f"Failed to load image {image_path}")
13             continue
14
15         # Convert the image to grayscale for face detection
16         gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
17
18         # Load the pre-trained Haar Cascade classifier for face detection
19         face_cascade = cv2.CascadeClassifier(cv2.data.harcascades + 'haarcascade_frontalface_default.xml')
20
21         # Detect faces in the image
22         faces = face_cascade.detectMultiScale(gray_img, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))
23
24         # Draw rectangles around the detected faces
25         for (x, y, w, h) in faces:
26             cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)
27
28         # Resize image to a fixed size
29         resized_img = cv2.resize(img, target_size)
30
31         # Plot input image with bounding boxes around detected faces
32         plt.figure(figsize=(5, 5))
33         plt.imshow(cv2.cvtColor(resized_img, cv2.COLOR_BGR2RGB))
34         plt.title(f'Input Image {i} with face detection')
35         plt.axis('off')
36         plt.show()
37
38 # Example usage:
39 image_paths = ["/content/baby image.jpeg", "/content/girlimage.jpeg", "/content/grandimage.jpeg", "/content/littleboy.jpg"]
40 detect_faces(image_paths)
41
```

Input Image 1 with face detection



Input Image 2 with face detection



Input Image 3 with face detection



```

1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 # Function to apply sharpening filter
6 def apply_sharpening(img):
7     # Apply sharpening filter to the original image
8     kernel_sharpening = np.array([[ -1, -1, -1], [-1, 9, -1], [-1, -1, -1]])
9     sharpened_img = cv2.filter2D(img, -1, kernel_sharpening)
10
11     return sharpened_img
12
13 # Function to resize images to a fixed size
14 def resize_image(img, target_size=(300, 300)):
15     return cv2.resize(img, target_size)
16
17 # Function to process images and apply sharpening
18 def process_images_sharpen(image_paths, target_size=(300, 300)):
19     plt.figure(figsize=(20, 10))
20
21     for i, image_path in enumerate(image_paths, start=1):
22         # Read the image using OpenCV
23         img = cv2.imread(image_path)
24
25         if img is None:
26             print(f"Failed to load image {image_path}")
27             continue
28
29         # Resize image to a fixed size
30         resized_img = resize_image(img, target_size)
31
32         # Apply sharpening filter
33         sharpened_img = apply_sharpening(resized_img)
34
35         # Plot sharpened image
36         plt.subplot(1, len(image_paths), i)
37         plt.imshow(cv2.cvtColor(sharpened_img, cv2.COLOR_BGR2RGB))
38         plt.title(f'Apply Sharpen Filter for Input {i}')
39         plt.axis('off')
40
41     plt.tight_layout()
42     plt.show()
43

```

```

1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 # Function to apply Gaussian blur
6 def apply_blur(img):
7     # Apply Gaussian blur to the original image
8     blurred_img = cv2.GaussianBlur(img, (25, 25), 0)
9
10     return blurred_img
11
12 # Function to resize images to a fixed size
13 def resize_image(img, target_size=(300, 300)):
14     return cv2.resize(img, target_size)
15
16 # Function to process images and apply blur
17 def process_images_blur(image_paths, target_size=(300, 300)):
18     plt.figure(figsize=(20, 10))
19
20     for i, image_path in enumerate(image_paths, start=1):
21         # Read the image using OpenCV
22         img = cv2.imread(image_path)
23
24         if img is None:
25             print(f"Failed to load image {image_path}")
26             continue
27
28         # Resize image to a fixed size
29         resized_img = resize_image(img, target_size)
30
31         # Apply blur filter

```