1️⃣

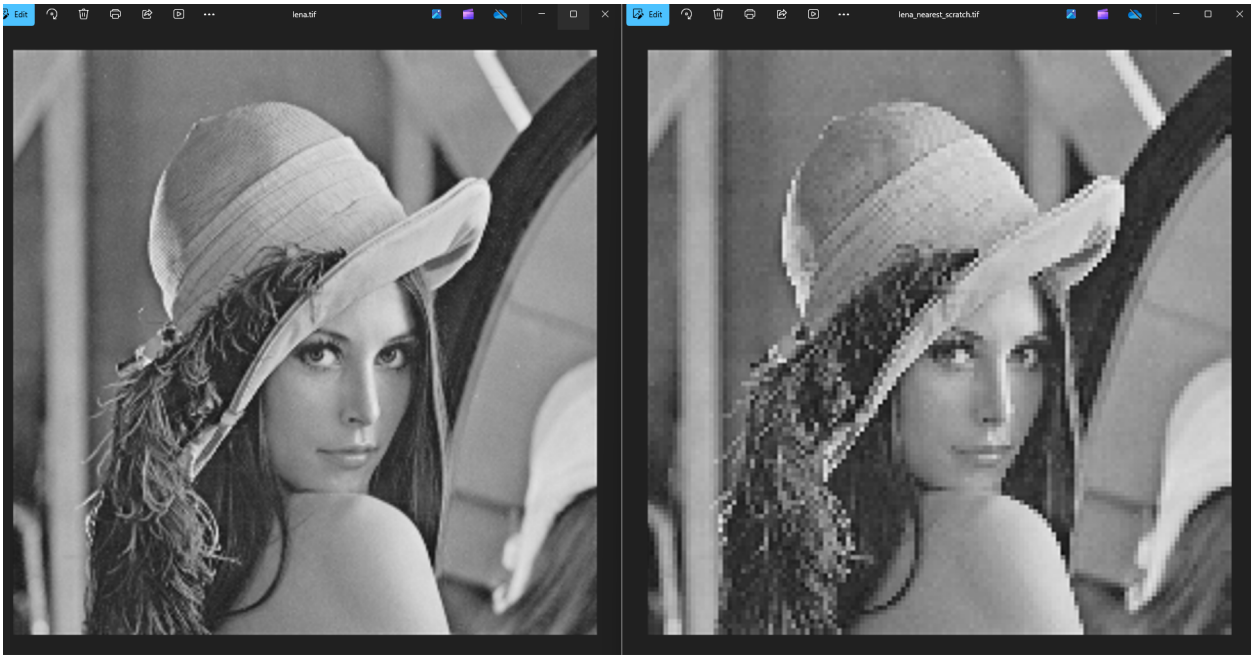
A1 - Jay Vora - Student # - 203321900

Class Name	CP467

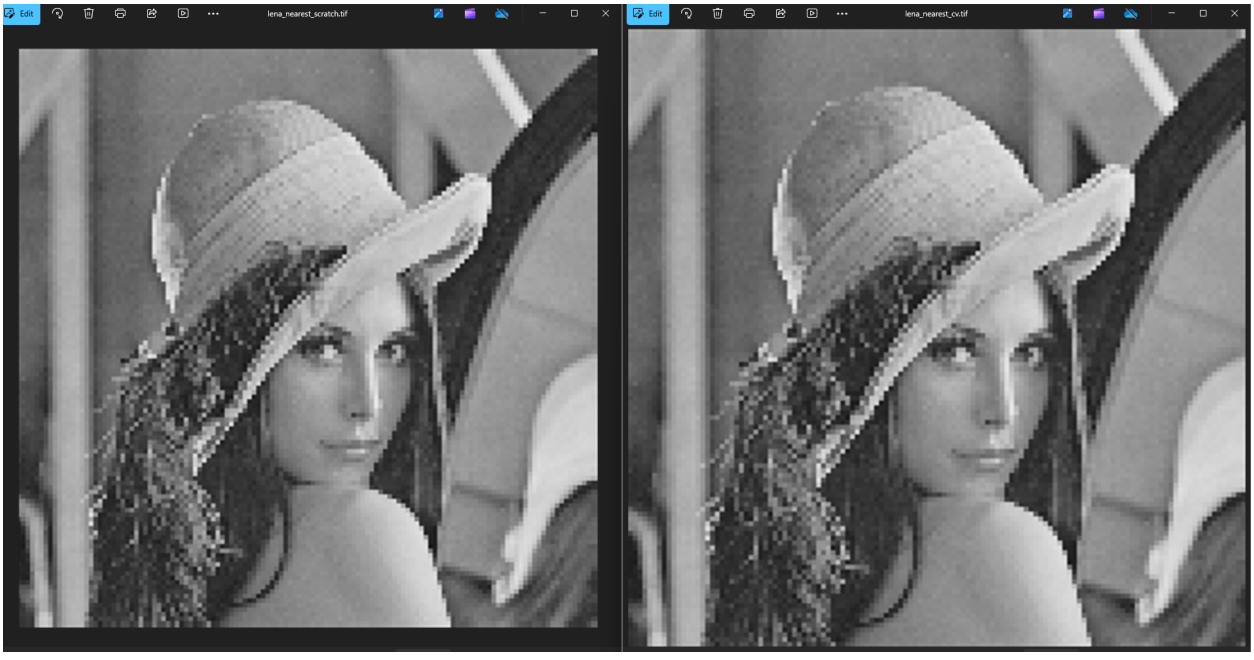
1) Image Interpolation

1. Nearest Neighbor

- the idea is for this interpolation technique is to have the nearest neighbor's pixel intensity as our own intensity.
- after getting new width and height through the scale factor we can obtain new coordinates by dividing the new pixel coordinates by the scaling factor and rounding to the nearest integer, which gives the corresponding pixel in the original image.
- This can be seen in both scaling down and scaling up scenarios as the image is seen to have more blurry lines as compared to the original image due to scaling down and scaling up using nearest neighbor interpolation.
- below is the example of original image and then scaled up image



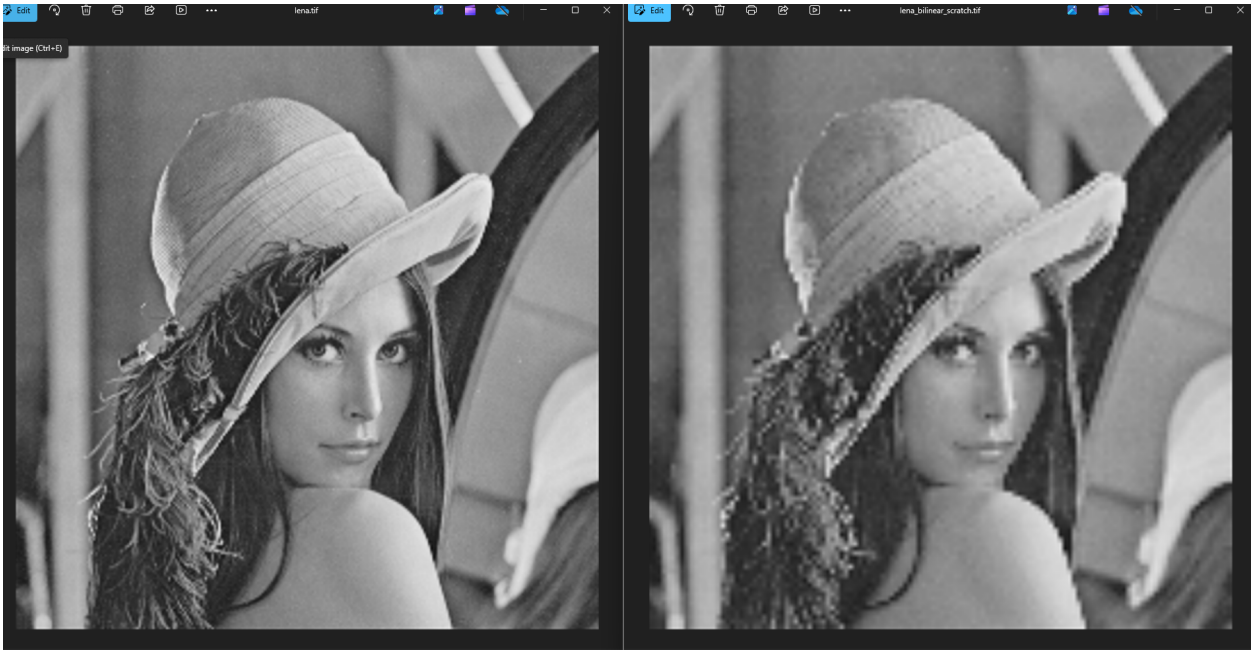
- Now comparing my own implementation with OpenCV's inbuilt method, there seems to be not much of difference as seen in below comparison.



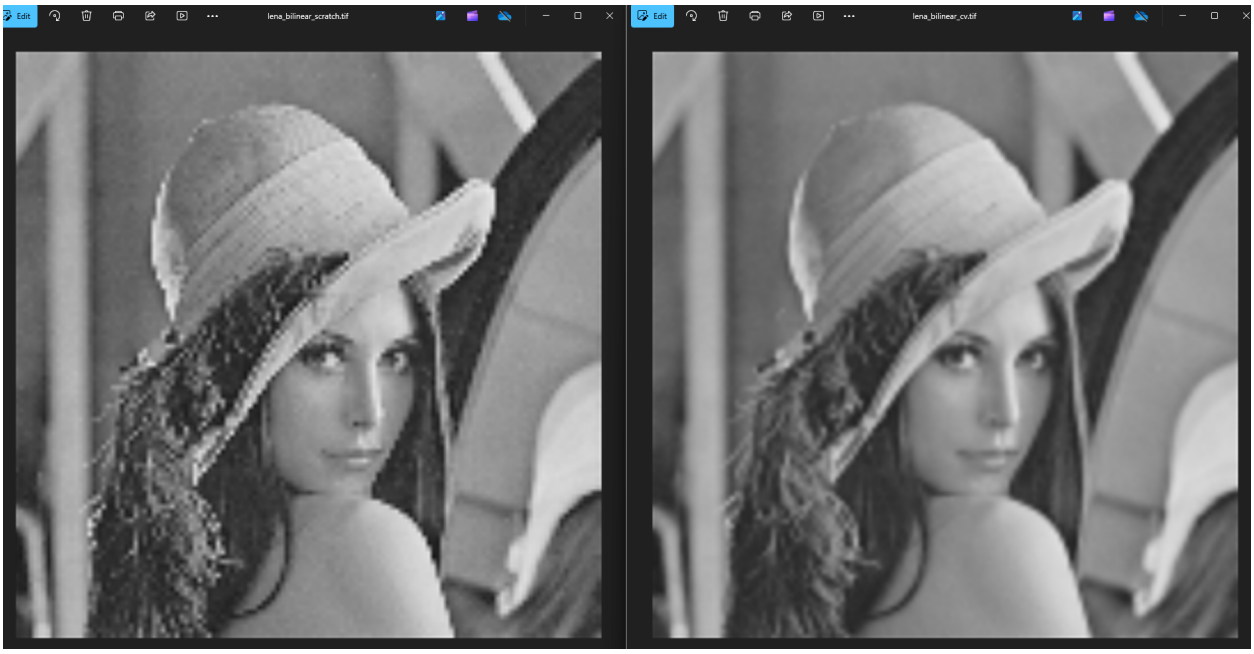
- Nearest neighbor is computationally efficient because of its simplicity but in terms of quality it doesn’t come close to bilinear and bicubic interpolation.

2. Bilinear Interpolation

- This technique uses the intensity values of four surrounding pixels to compute a new pixel value.
- In my implementation the new pixel value (considering the scaling up or down), it maps the pixel back to previous image and gathers nearest four pixels and then performs linear interpolation along x-axis twice and then followed by y-axis linear interpolation. This is done using the distances between new pixels and its neighbors.
- This technique gives smoother image compared to nearest neighbor but computationally not as efficient as that.
- Below is the comparison of the original image and bilinear_scratch (visibly the difference can be seen through the scratch image as it is more clearer than than the previous comparison.)



- Below id the comparison between bilinear scratch and OpenCV's bilinear (clearly the OpenCV's inbuilt method is more clear than my implementation.)



- I have had difficulties calculating the top and bottom interpolation and then tried to visualize it with a small 5×5 image to get the idea of how to create the equation and was able to overcome that difficulty by simply visualizing that.

3. Bicubic Interpolation

- This technique utilizes the values of the 16 nearest pixels surrounding the target pixel to achieve smoother and better image quality. The more precision, the better the picture quality.
- Below is the bilinear and bicubic comparison (bilinear can appear smoother but can be more blurred whereas better details and sharpness can be seen in the bicubic image)

2) Point Operations

1. Negative

- This technique involves inverting the intensity values of each pixel in an image.
- For greyscale image, this can be done by subtracting each pixel value from 255.
- The negative image provides a complete inversion of tones. So, original dark areas appear bright in the negative whereas bright areas appear dark.
- Below is the comparison b/w original and negative

2. Power-law Transformation

- This technique is a nonlinear technique used to adjust image contrast.
- Implementation steps include Normalization, Transformation and Denormalization
    - Normalization: Scale pixel values to range of [0,1]
    - Transformation: Apply power-law formula
    - Denormalization: Scale back to range of [0,255] and ensure the values are within bounds.
- Quality of results:
    1. **Gamma > 1**: Darkens the image

       for Gamma = 1.2

1. **Gamma < 1**: Brightens the image

   For Gamma = 0.1

1. **Gamma = 1**: No change (linear transformation)

   For Gamma = 1

- According to me darker image is the best output among these images which means **Gamma > 1** should give the best result for given cameraman image.

3. Bit-Plane Slicing

- This technique separates an image into individual bit planes, revealing the contribution of each bit to the overall image.
- For an 8-bit grayscale image, 8 bit planes from least significant bit to the most significant bit.
- My implementation iterates through each pixel and extracts the bit using bitwise operations and creates new image where extracted bit determines

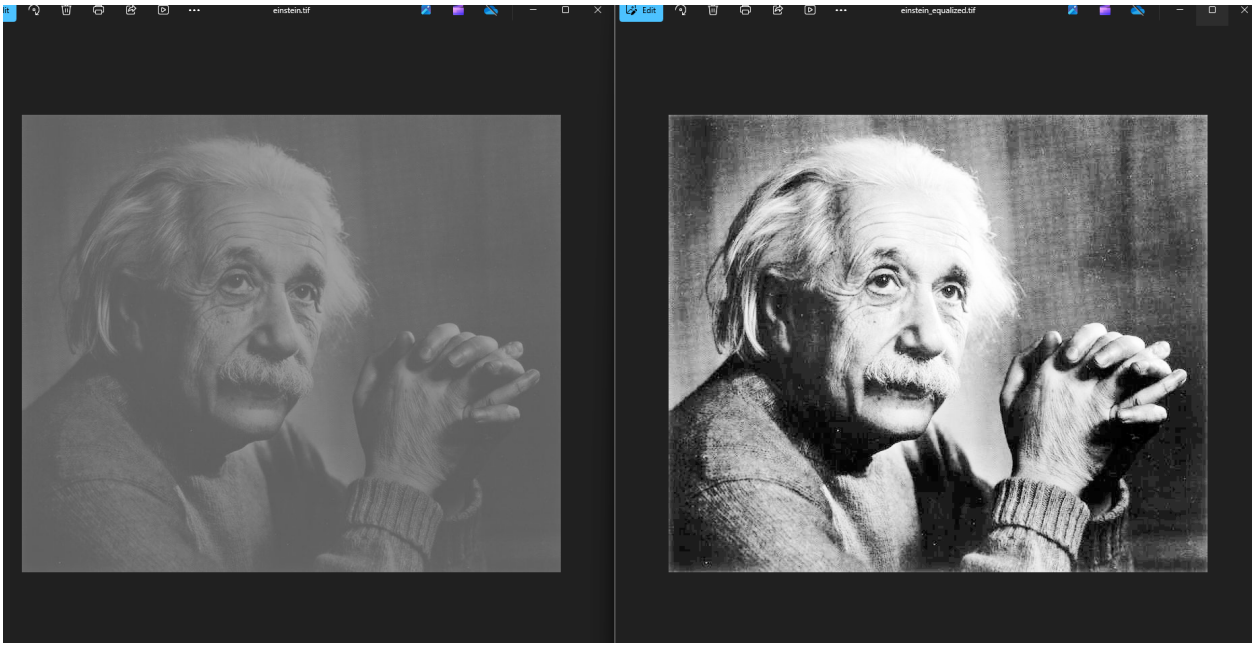
pixel intensity.

- Least significant bits are lower in quality meaning they have more noise than the most significant bits.
- The most important difficulty i face was to make changes so that the bitwise operations would work to get the significant bits from the 8-bit grayscale. Using right shift made the job easier after few google searches I was able to make it work so that the bit is gathered based on the bit plane.

3) Histogram Processing

1. Histogram Equalization

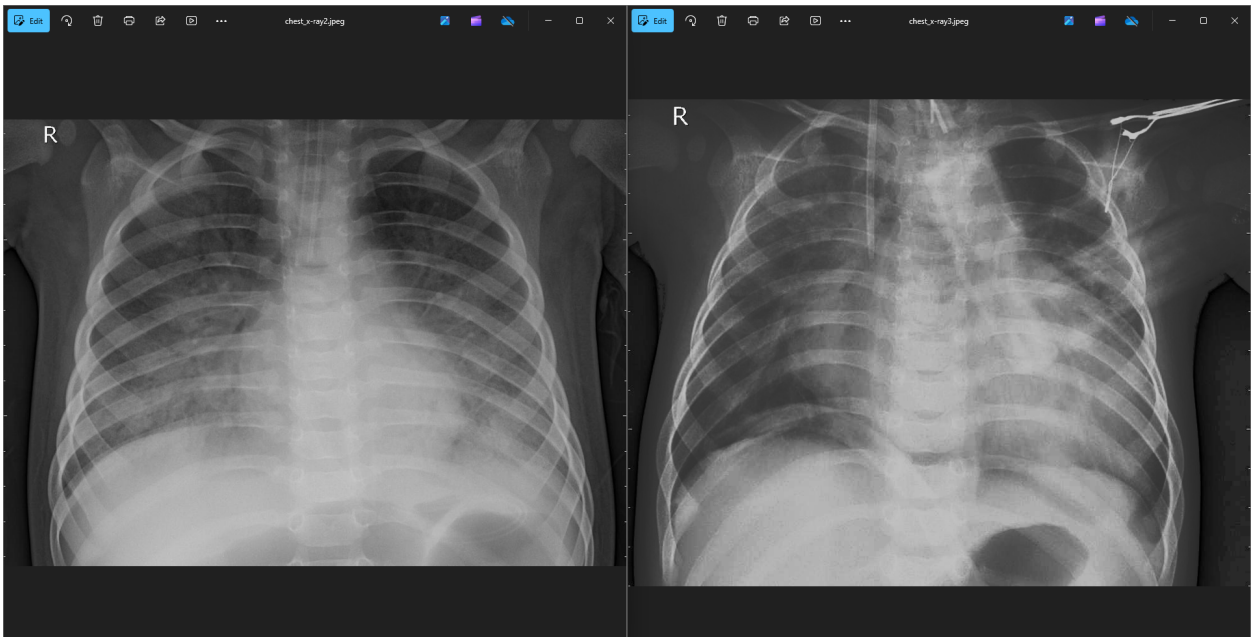
- This technique is to enhance image contrast by redistributing pixel intensities more uniformly across the available range.
    - Calculating the image histogram - occurrences of each intensity level
    - Calculating the cumulative distribution function (CDF) - computes and normalizes the cumulative sum of the histogram
    - Normalizing the CDF to map original pixel values to new more evenly distributed values.
- Below is the comparison between original and equalized image (as it can be seen that equalized image shows increased contrast, revealing more details. Also, visibility is also increased.)



- Getting the CDF normalized was one of the difficult part of this task to achieve as needed to have the pixels intensities int he range of 0 to 255. by performing that step values are evenly distributed across the range of 0 to 255.

2. Histogram Specification

- This technique transforms one image to have a histogram similar to a target image.
- Below is the implementation
  - Calculate histogram
  - Calculate the CDF
  - Create mapping between input and target intensity levels - Find closest matches between input and target CDFs.
  - Applying the mapping to the input image - Transform input image using the created mapping
- Below is the comparison between target and the output image. (it can clearly be noticed that input image had some noise on left side of the image and that has carried into the third image that means that result may not exactly the same as the target image.)


Edit
chest_x-ray2.jpeg
R
Edit
chest_x-ray3.jpeg
R