CS 415
Final Project Report

Denoising Chest X-ray Images

Hossein Fathollahian
Mitravinda Manjunath
Shivam Moudgil
Chirag Deepak Shinde
Sushanth Thota

University of Illinois at Chicago
Fall 2024

Contents

Abstract
Medical imaging is crucial for diagnosing and treating diseases, with chest X-rays being widely used. However, noise in these images degrades quality, hindering accurate diagnosis. Effective denoising techniques are necessary to enhance clarity while preserving diagnostic details. This project implemented and evaluated traditional filters and deep learning methods for denoising chest X-ray images. Traditional techniques included Mean, Median, Gaussian, Bilateral, Wiener, Wavelet, and Fourier filters. Deep learning approaches used are Generative Adversarial Networks (GANs), Autoencoders, and U-Net architectures. Performance was assessed using the Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM). The Autoencoder showed the best performance, providing higher noise reduction and structural preservation (PSNR = 29.441[†], SSIM = 0.801) compared to other methods.

Keywords: Chest X-ray Image, Denoising, Traditional Filters, Deep Learning Models

---

1 Introduction

Medical images are indispensable in the diagnosis and treatment of numerous medical conditions, providing critical insights into a patient's internal structures and abnormalities[10]. Among these, chest X-rays are one of the most commonly utilized imaging technologies, capturing detailed images of the lungs, heart, airways, blood vessels, and the bones of the spine and chest. However, interpreting chest X-ray images can be challenging due to the presence of noise and artifacts, which degrade image quality during transmission and acquisition. This further complicates the diagnosis and treatment process[6]. Thus, to improve the quality of the image, denoising is a critical issue in the field of image processing.

Some common types of image noise are: Gaussian noise, Poisson noise, Impulse noise, Speckle noise and Salt and Pepper noise[11;1]. As a fundamental step of image processing, image denoising needs to remove the noise and preserve image details. Broadly, the techniques for denoising images can be categorized into: traditional methods and deep learning methods[9]. In the case of traditional methods, popular filters include Mean, Median, Gaussian, Bilateral, Wiener, Wavelet and Fourier denoising filters. It is often challenging to have prior knowledge of the specific type of noise present in an image during the denoising process. Deep learning approaches are inherently more robust and excel in handling varying and unknown noise types. Their remarkable ability to learn complex, high-dimensional patterns from large datasets allows them to adapt to different noise conditions without the need for explicit prior knowledge. By leveraging deep neural networks, these methods can effectively distinguish between noise and meaningful image content, enabling them to remove noise with high precision while minimizing image distortion[9].

[†]When 24-bit depth images were used, a PSNR of 55 was obtained for the autoencoder model.

2 Goal and Problem Statement

2.1 Goal

The goal of this study is to address the challenges with medical images, specifically chest X-ray images, due to degradation by noise and artifacts introduced during the acquisition and transmission processes. The project focuses on enhancing the quality of chest X-ray images by implementing and evaluating various denoising techniques. This ultimately helps in contributing to improving medical image quality and facilitating more accurate diagnosis and better patient outcomes.

2.2 Problem Statement

- To implement different traditional filters, such as Mean, Median, Gaussian, Bilateral, Weiner, Wavelet, and Fourier Denoising; compare against advanced deep learning models, including Generative Adversarial Networks (GAN), Autoencoders, and U-Net architectures.
- Additionally, to assess the performance of these methods using metrics such as Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM), identifying the most effective approach for denoising chest X-ray images.

3 Dataset

This project uses the ChestX-ray pneumonia dataset[7], which consists of X-ray images of healthy and pneumonia-affected lungs. Various types of noise were introduced into the original X-ray images to simulate real-world conditions and enhance the robustness of the denoising models. These noise types mimic distortions commonly encountered during image acquisition, transmission, or storage, replicating practical challenges faced in medical imaging. The following types of noise were applied:

1. **Gaussian Noise:** It adds random variations in pixel intensity according to Gaussian distribution. Usually, this is an electronic sensor noise.
2. **Impulse Noise:** It creates sudden, sharp disturbances in pixel values that are normally due to errors in the data transmission or background sensor flaws.
3. **Poisson Noise:** It is also known as shot noise. It models the statistical variations in the number of photons detected during imaging.
4. **Salt and Pepper Noise:** It randomly infuses black (pepper) and white (salt) pixels to the image.
5. **Speckle Noise:** It is a granular noise that degrades the image-quality by injecting multiplicative distortions.

The above noises are introduced to the original (clean) images in equal ratio. Figure 1 shows a snapshot of the original and noisy chest X-ray images.

1. **Training Dataset:** The training dataset consists of a total of 2100 images of clean X-rays and 2100 images of noisy X-rays. This dataset is used to train the deep learning models for the task of denoising.
2. **Test Dataset:** The testing dataset consists of total of 450 images of clean X-rays and 450 images of noisy X-rays. This dataset is used to test the performance of traditional filters and the trained deep learning models.

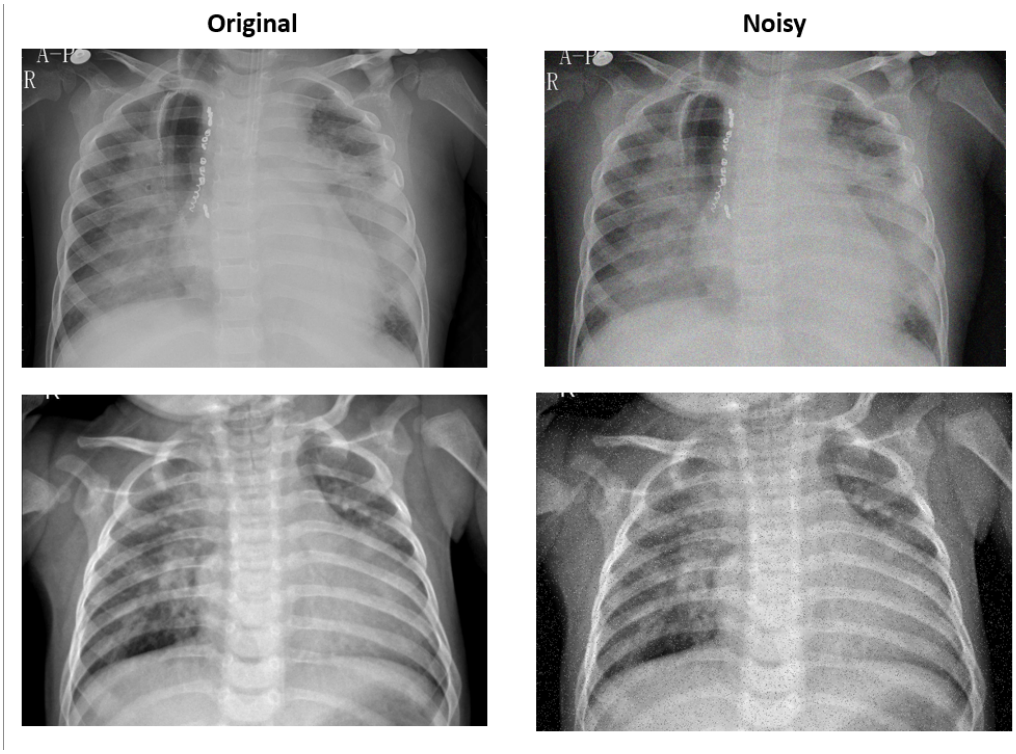

Figure 1: Dataset Overview

4 Methodology

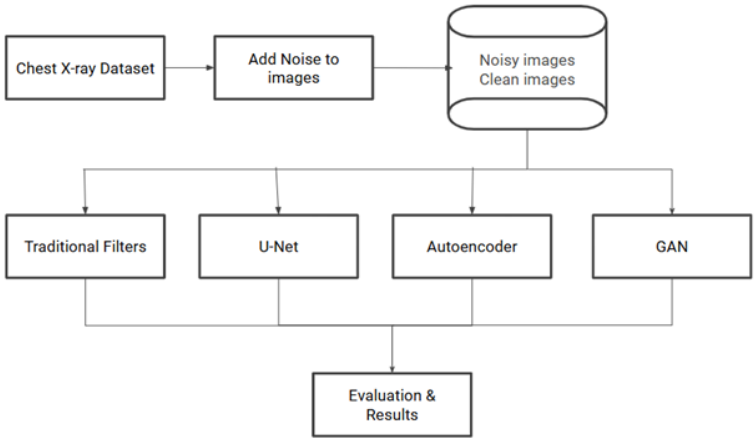

Figure 2: Project Methodology Overview

Figure 2 provides an overview of the implementation process adopted in this project. Initially, five different types of noises were introduced into the existing dataset of chest X-ray images namely, Gaussian, Impulse, Salt and Pepper, Poisson and Speckle noise. Subsequently, various traditional filtering techniques and deep learning models were applied to restore the images and reduce noise. Finally, the performance of each method was evaluated and compared through comprehensive analysis, using metrics such as PSNR and SSIM to determine the most effective denoising approach.

4.1 Traditional filters

Various traditional filters were implemented to enhance the chest X-ray images by removing noises without distorting the features. The methods employed here include Mean, Median, Gaussian, Bilateral, Weiner, Wavelet, and Fourier Denoising filters.

- **Mean Filter**: Mean or average filtering technique reduces noise by averaging the pixel values within a local neighborhood, producing a smoother image but possibly blurring edges. The filter can be described as:

$$G(x,y) = \frac{1}{(2k+1).(2k+1)} \sum_{i=-k}^{k} \sum_{j=-k}^{k} f(x+i, y+j)$$

  where, $G(x,y)$ is the filtered pixel value,
  $f(x+i, y+j)$ is the pixel values in the neighborhood,
  k is the neighborhood size

- **Median Filter**: Median filtering replaces each pixel value with the median of its neighborhood, effectively removing salt-and-pepper noise while preserving edges. The median filter can be described using the below equation:

$$G(x,y) = \texttt{Median}\{f(x+i, y+j)| - k \leq i, j \leq k\}$$

  where, $G(x,y)$ is the filtered pixel value,
  `Median` is the median value of the neighborhood,
  k is the neighborhood size

- **Gaussian Filter**: Gaussian filtering applies a weighted average based on a Gaussian kernel, reducing noise while slightly smoothing edges. It can be given by:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

  where, $G(x,y)$ is the filtered pixel value,
  σ is the standard deviation of the Gaussian kernel,
  x, y are the spatial coordinates

- **Bilateral Filter**: Bilateral filtering smooths images while preserving edges by considering both spatial and intensity differences. The filter can be described as:

$$G(x,y) = \frac{1}{W_p} \sum_{i=-k}^{k} \sum_{j=-k}^{k} f(x+i, y+j) \cdot e^{-\frac{i^2+j^2}{2\sigma_s^2}} \cdot e^{-\frac{(f(x,y)-f(x+i,y+j))^2}{2\sigma_r^2}}$$

where, $G(x, y)$ is the filtered pixel value,
σ_s is the spatial smoothing parameter,
σ_r is the intensity smoothing parameter,
W_p is the normalization factor,
k is the neighborhood size

- **Wiener Filter**: Wiener filtering adapts to local variance in the image, aiming to minimize the mean square error between the original and filtered image. It can be described as:

$$G(u, v) = \frac{H^*(u, v)|F(u, v)|^2}{|H(u, v)|^2|F(u, v)|^2 + K} F(u, v)$$

where, $G(x, y)$ is the filtered pixel value,
$H(u, v)$ is the degradation function,
$F(u, v)$ is the Fourier transform of the image,
$H^*(u, v)$ is the Complex conjugate of $H(u, v)$,
K is the Noise-to-signal ratio constant

- **Wavelet Filter**: Wavelet denoising decomposes the image into wavelet coefficients and removes noise by thresholding in the frequency domain. It can be explained as:

$$W_\psi(j, k) = \sum_{n=0}^{N-1} f(n)\psi_{j,k}(n)$$

where, $W_\psi(j, k)$ is the Wavelet coefficient,
$\psi_{j,k}(n)$ is the Wavelet basis functions,
$f(n)$ is the Image signal,
N is the Signal length

- **Fourier Filter**: Fourier denoising removes high-frequency noise by transforming the image to the frequency domain and filtering undesired frequencies.

$$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-2\pi\iota\left(\frac{ux}{M} + \frac{vy}{N}\right)}$$

where, $F(u, v)$ is the Fourier transform of the image,
$f(x, y)$ refer to the pixel values in the spatial domain,
M, N are the dimensions of the image,
ι is the imaginary unit

4.2 U-Net

U-Net is a deep learning architecture based on Convolutional Neural Networks (CNN). It is made up of encoder layers and decoder layers arranged in contracting and expansive paths of the network respectively. In U-Net, the encoder and decoder parts are symmetrical and the layers of the encoder are linked to the corresponding decoder layers through skip connections.
Encoder: Feature Extraction
The encoder of U-Net captures key features from the input image using convolutional and pooling layers. Each stage performs convolutional operations and decrease the spatial dimensions

along with increasing the depth, typically using max-pooling.

Decoder and Skip Connection:

The layers of the decoder perform convolutional operations and upsample the feature maps to decode the encoded data. This way, it reconstructs image resolution. At each stage, the decoder combines with its features, the information from the skip connections. Skip connection creates a link between the encoder path (contracting path) and feature maps in the decoder path (expanding path). It helps to preserve the spatial information lost in the contracting path.

4.2.1 Network Architecture

Figure 3 sumarizes the architeture of U-Net used in this project.

Encoder Architecture:

The input to the model is a grayscale chest X-ray image of shape 128×128×1. The encoder has 4 blocks. Each block includes 2 convolutional layers (kernel size: 3x3) followed by a max-pooling layer (kernel size: 2x2), progressively reducing the spatial dimenions of the features and increasing the number of channels from 64, 128, 256 to 512. ReLU activation function is used in the convolutional layers.

Decoder Architecture:

The decoder consists of 4 blocks. Each block contains 2x2 upsampling layer, to double the spatial dimensions, followed by 2 convolutional layers. The skip connections concatenate the feature maps from the corresponding encoder layer to the decoder layer at the same level. This allows the network to recover spatial details lost during down-sampling. ReLU activation function is used in the convolutional layers.

4.2.2 Model Training

The U-Net model was trained for 20 epochs with a batch size of 16. The learning rate was set to 0.001. Mean squared error was used as the loss function along with Adam optimizer to train the encoder and decoder parts of the network.

Model: "functional"

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 128, 128, 1)	0	-
conv2d (Conv2D)	(None, 128, 128, 64)	640	input_layer[0][0]
conv2d_1 (Conv2D)	(None, 128, 128, 64)	36,928	conv2d[0][0]
max_pooling2d (MaxPooling2D)	(None, 64, 64, 64)	0	conv2d_1[0][0]
conv2d_2 (Conv2D)	(None, 64, 64, 128)	73,856	max_pooling2d[0][0]
conv2d_3 (Conv2D)	(None, 64, 64, 128)	147,584	conv2d_2[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 32, 32, 128)	0	conv2d_3[0][0]
conv2d_4 (Conv2D)	(None, 32, 32, 256)	295,168	max_pooling2d_1[0][0]
conv2d_5 (Conv2D)	(None, 32, 32, 256)	590,080	conv2d_4[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 16, 16, 256)	0	conv2d_5[0][0]
conv2d_6 (Conv2D)	(None, 16, 16, 512)	1,180,160	max_pooling2d_2[0][0]
conv2d_7 (Conv2D)	(None, 16, 16, 512)	2,359,808	conv2d_6[0][0]
conv2d_transpose (Conv2DTranspose)	(None, 32, 32, 256)	1,179,904	conv2d_7[0][0]
concatenate (Concatenate)	(None, 32, 32, 512)	0	conv2d_transpose[0][0… conv2d_5[0][0]
conv2d_8 (Conv2D)	(None, 32, 32, 256)	1,179,904	concatenate[0][0]
conv2d_9 (Conv2D)	(None, 32, 32, 256)	590,080	conv2d_8[0][0]
conv2d_transpose_1 (Conv2DTranspose)	(None, 64, 64, 128)	295,040	conv2d_9[0][0]
concatenate_1 (Concatenate)	(None, 64, 64, 256)	0	conv2d_transpose_1[0]… conv2d_3[0][0]
conv2d_10 (Conv2D)	(None, 64, 64, 128)	295,040	concatenate_1[0][0]
conv2d_11 (Conv2D)	(None, 64, 64, 128)	147,584	conv2d_10[0][0]
conv2d_transpose_2 (Conv2DTranspose)	(None, 128, 128, 64)	73,792	conv2d_11[0][0]
concatenate_2 (Concatenate)	(None, 128, 128, 128)	0	conv2d_transpose_2[0]… conv2d_1[0][0]
conv2d_12 (Conv2D)	(None, 128, 128, 64)	73,792	concatenate_2[0][0]
conv2d_13 (Conv2D)	(None, 128, 128, 64)	36,928	conv2d_12[0][0]
conv2d_14 (Conv2D)	(None, 128, 128, 1)	65	conv2d_13[0][0]

Figure 3: U-Net Network Architecture

4.3 GAN

Generative Adversarial Network (GAN) is a neural network architecture. It consists of two neural networks that compete against each other: the Generator (G) and the Discriminator (D). The main goal of GAN is to learn the patterns from the given data using which it can generate new data close to the original data.

Generator:
The generator is a neural network model that takes in random noise as input and tries to generate synthetic images that closely match the original training data. If z is a random noise vector sampled from some noise distribution $P_z(z)$, the output of the image by the generator can be given by $G(z)$. It is trained to learn the underlying distribution of the training images and tries to fool the discriminator, whose goal is to differentiate between real and synthetic images. For the task of denoising, the GAN model typically contains sequential downsampling and upsampling convolutional layers, and data normalization and activation functions such as Leaky ReLU are utilized. This allows the generator to learn to map noisy input images to denoised output images.

```
Generator Model Summary:
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
Generator                                [1, 1, 64, 64]            --
├─Sequential: 1-1                        [1, 1, 64, 64]            --
│    └─Conv2d: 2-1                       [1, 64, 32, 32]           1,088
│    └─LeakyReLU: 2-2                    [1, 64, 32, 32]           --
│    └─Conv2d: 2-3                       [1, 128, 16, 16]          131,200
│    └─BatchNorm2d: 2-4                  [1, 128, 16, 16]          256
│    └─LeakyReLU: 2-5                    [1, 128, 16, 16]          --
│    └─Conv2d: 2-6                       [1, 256, 8, 8]            524,544
│    └─BatchNorm2d: 2-7                  [1, 256, 8, 8]            512
│    └─LeakyReLU: 2-8                    [1, 256, 8, 8]            --
│    └─ConvTranspose2d: 2-9              [1, 128, 16, 16]          524,416
│    └─BatchNorm2d: 2-10                 [1, 128, 16, 16]          256
│    └─ReLU: 2-11                        [1, 128, 16, 16]          --
│    └─ConvTranspose2d: 2-12             [1, 64, 32, 32]           131,136
│    └─BatchNorm2d: 2-13                 [1, 64, 32, 32]           128
│    └─ReLU: 2-14                        [1, 64, 32, 32]           --
│    └─ConvTranspose2d: 2-15             [1, 1, 64, 64]            1,025
│    └─Tanh: 2-16                        [1, 1, 64, 64]            --
------------------------------------------------------------------------------------------
Discriminator Model Summary:
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
Discriminator                            [1, 1]                    --
├─Sequential: 1-1                        [1, 1]                    --
│    └─Conv2d: 2-1                       [1, 64, 32, 32]           1,088
│    └─LeakyReLU: 2-2                    [1, 64, 32, 32]           --
│    └─Conv2d: 2-3                       [1, 128, 16, 16]          131,200
│    └─BatchNorm2d: 2-4                  [1, 128, 16, 16]          256
│    └─LeakyReLU: 2-5                    [1, 128, 16, 16]          --
│    └─Conv2d: 2-6                       [1, 256, 8, 8]            524,544
│    └─BatchNorm2d: 2-7                  [1, 256, 8, 8]            512
│    └─LeakyReLU: 2-8                    [1, 256, 8, 8]            --
│    └─Flatten: 2-9                      [1, 16384]                --
│    └─Linear: 2-10                      [1, 1]                    16,385
│    └─Sigmoid: 2-11                     [1, 1]                    --
==========================================================================================
Total params: 673,985
Trainable params: 673,985
Non-trainable params: 0
Total mult-adds (M): 68.29
==========================================================================================
```

Figure 4: GAN Network Architecture

Discriminator:
The discriminator model of GAN is a neural network that is trained to differentiate between the ground truth images x and the generated or synthesized images $G(z)$. Thus, the discriminator functions as a binary classifier by evaluating the input images and assigning a probability of authenticity to each input. The discriminator, D is typically a convolutional network, outputting a probability map indicating whether image regions are real or fake.

These Generator (G) and Discriminator (D) networks are trained alternatively until the generated images are almost indistinguishable from the original images. The objective function for this process is given by:

$$\min_G \max_D E_{x \sim p_{\text{data}}(x)} [\log D(x)] + E_{z \sim p_z(z)} [\log(1 - D(G(z)))] \tag{1}$$

where G is the generator,
D is the discriminator,

$P_{data}(x)$ is the real data distribution,
$P_z(z)$ is the prior distribution on the noise vector z,
$G(z)$ is the generated data from noise vector z,
$D(x)$ is the probability that x is real (i.e., from the real data distribution).

4.3.1 Network architecture

Figure 4 summarizes GAN's network architecture.
Generator Architecture:
It utilizes a series of convolutional, transposed convolutional, and normalization layers to progressively encode and decode the input features, maintaining spatial details while reducing noise.

- Input and Initial Layers: The input image of size 64×64 is processed through a sequence of convolutional layers with increasing filter depths (64, 128, and 256) and stride-2 downsampling, effectively capturing multi-scale spatial features. LeakyReLU activations is used in these layers.
- Bottleneck: After progressively downsampling the input, the latent representation is formed which captures the essential features required for denoising.
- Upsampling Layers: Transposed convolutional layers are applied to reconstruct the original image size while reducing noise. The filter depths are decreased in reverse order (256 $\rightarrow$ 128 $\rightarrow$ 64 $\rightarrow$ 1).
  Batch normalization layers after each transposed convolution stabilize learning and improve generalization. ReLU activation function is employed in the upsampling path and a Tanh function is used at the final layer to scale the output between -1 and 1.
- Output: The Generator outputs a denoised chest X-ray image of the same size as the input (64×64).

Discriminator Architecture:

- Input and Convolutional Layers: The Discriminator begins with convolutional layers to extract hierarchical features from the input. The filter depths increase progressively (64 $\rightarrow$ 128 $\rightarrow$ 256), enabling the model to capture complex patterns. Batch normalization and LeakyReLU activations are used to ensure stable training and effective handling of non-linear features.
- Flattening and Classification: After feature extraction, the output is flattened into a single-dimensional vector and passed through a fully connected linear layer. The final Sigmoid activation outputs a probability score, indicating whether the input image is real or generated.

4.3.2 Model Training

The training process for GAN spanned 25 epochs with a batch size of 16. The learning rate was set to 0.0002, ensuring gradual updates to the model's weights for stable learning. The Adam optimizer was employed to manage the dynamic learning rates during training. The Binary Cross Entropy (BCE) loss function was used to train both the generator and discrimi-

nator, ensuring the generator produced denoised images close to the original images, while the discriminator effectively differentiated between denoised and ground truth images.

4.4 Autoencoder

Autoencoder is a deep learning architecture. It mainly learns 2 functions- the encoding function and the decoding function. The encoder transforms the input data by compressing it down to its essential features. The decoder on the other hand reconstructs the original data from the compressed representation.

Encoder:
The encoder compresses or encodes the input image into a low-dimensional representation - latent space. This part of the network reduces the image's spatial dimensions and focuses on capturing essential features. Typically, encoder comprises of convolutional layers followed by pooling layers to downsample the input image.

$$\mathbf{z} = f(\mathbf{x}; \theta_e)$$

Where:

- $\mathbf{x}$ is the input vector (e.g., an image, text, etc.),
- $f(\mathbf{x}; \theta_e)$ represents the encoder network, parameterized by θ_e,
- $\mathbf{z}$ is the latent (encoded) representation.

Decoder:
The decoder reconstructs the clean image from the latent representation. This part of the network gradually upsamples the encoded features back to the original input size. Typically, transposed convolutions or upsampling layers are used to reverse the downsampling process.

$$\hat{\mathbf{x}} = g(\mathbf{z}; \theta_d)$$

Where:

- $\mathbf{z}$ is the latent representation obtained from the encoder,
- $g(\mathbf{z}; \theta_d)$ is the decoder network, parameterized by θ_d,
- $\hat{\mathbf{x}}$ is the reconstructed output (an approximation of the input).

Final Autoencoder Training Objective: Objective is to minimise the loss as

$$\min_{\theta_e, \theta_d} \mathcal{L}(\mathbf{x}, \hat{\mathbf{x}})$$

4.4.1 Network Architecture

The autoencoder is a symmetrical CNN with an encoder, a bottleneck, and a decoder. Fig 5 shows the architecture used by us. Steps involved are:

- **Input**: Model receives 256x256x1 input image. Here 1 represents the channel, which means single channel image.

Layer (type)	Output Shape	Param #
image_input (InputLayer)	(None, 256, 256, 1)	0
conv2d_12 (Conv2D)	(None, 256, 256, 64)	640
batch_normalization_15 (BatchNormalization)	(None, 256, 256, 64)	256
max_pooling2d_6 (MaxPooling2D)	(None, 128, 128, 64)	0
conv2d_13 (Conv2D)	(None, 128, 128, 128)	73,856
batch_normalization_16 (BatchNormalization)	(None, 128, 128, 128)	512
max_pooling2d_7 (MaxPooling2D)	(None, 64, 64, 128)	0
conv2d_14 (Conv2D)	(None, 64, 64, 256)	295,168
batch_normalization_17 (BatchNormalization)	(None, 64, 64, 256)	1,024
conv2d_transpose_6 (Conv2DTranspose)	(None, 128, 128, 128)	295,040
batch_normalization_18 (BatchNormalization)	(None, 128, 128, 128)	512
conv2d_transpose_7 (Conv2DTranspose)	(None, 256, 256, 64)	73,792
batch_normalization_19 (BatchNormalization)	(None, 256, 256, 64)	256
conv2d_15 (Conv2D)	(None, 256, 256, 1)	577

Figure 5: Autoencoder Architecture

- **Encoder Architecture:**
  We have used 3 convolutional layers at encoder. The number of the kernel filters (size: 3x3) used in each layer is 64, 128 and 256 respectively. Convolutional layers help in extracting spatial features from the input image by applying these kernel filters. Each convolutional layer uses ReLU activation function along with Batch Normalization. Additionally, 2 max-pooling layers are used to downsample the images while preserving only the most significant features .This creates a compressed representation of the input image.
- **Bottleneck:** At the bottleneck, the spatial dimensionality of the image is the most compressed - dimensions of (64, 64) and 256 feature maps. This step enables the model to focus on the most important features for reconstructing the denoised image.
- **Decoder Architecture:**
  Decoder consists of 2 transpose convolution layers, These are used to upsample the the compressed image to its original size of 256x256. ReLU was used as the activation function along with Batch Normalization.
- **Output Layer:** A final Conv2D layer with sigmoid activation outputs the denoised image [0, 1].

4.4.2 Model Training

The autoencoder model was trained for 20 epochs with images in batches of 16. During the training process, the encoder and decoder parts were trained to optimize the Mean Squared Error (MSE) loss function. Adam optimizer was utililized and the learning rate was set at 0.0001.

5 Results

The performance of all the traditional filters and deep learning models was tested using the test dataset of chest X-ray images (containing 450 images). The following sections detail on the performance of each denoising method implemented.

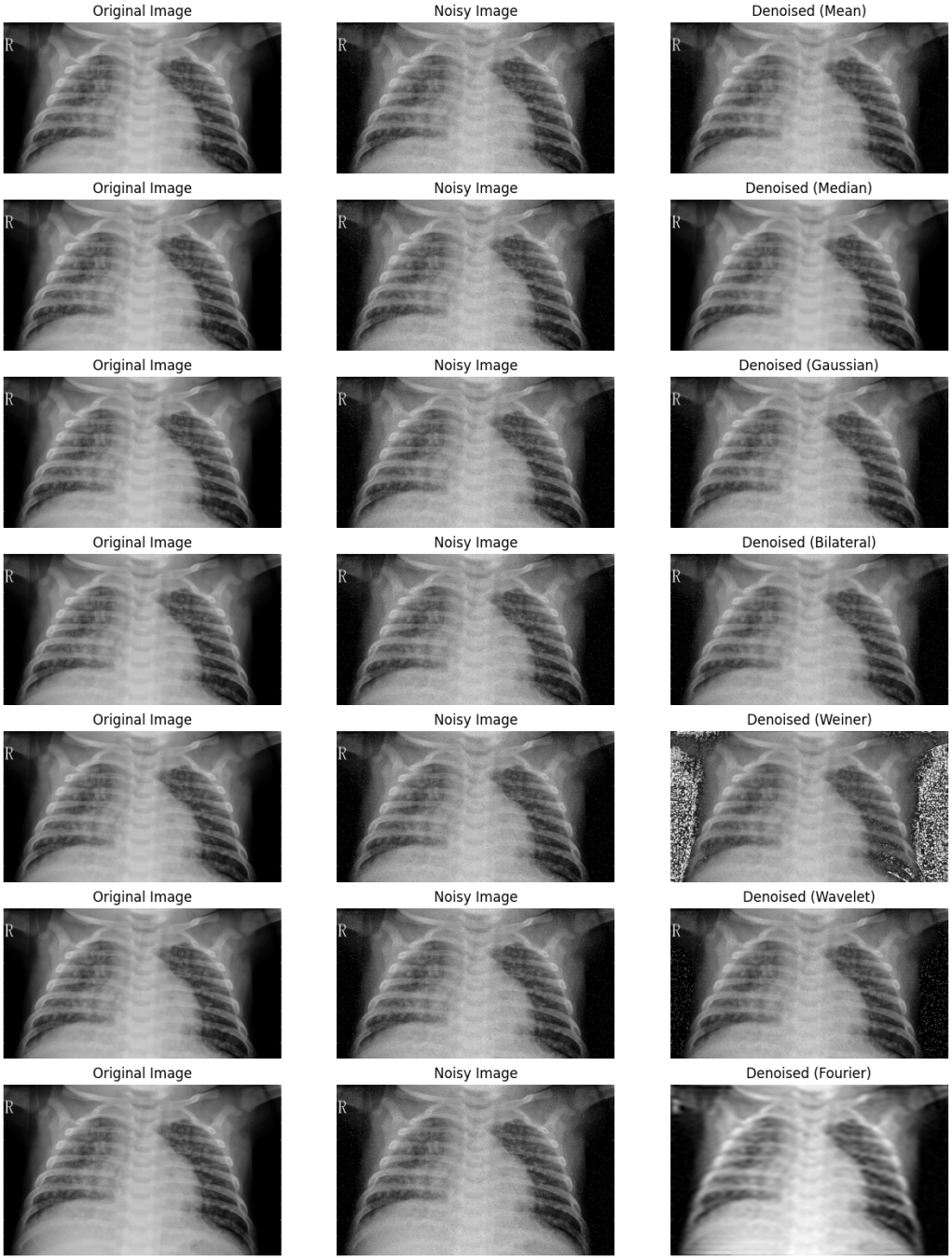

Figure 6: Traditional-Filters Denoising Results

5.1 Traditional Filters

Figure 6 shows the results of denoising via different traditional filters. Each comparison features three key images: the original image (ground truth), the noisy version of the image, and denoised image (output of the particular traditional filter). There are seven subfigures in fig 6. Each subfigure is associated with the corresponding traditional filter used - Mean, Median, Gaussian, Bilateral, Wiener, Wavelet, and Fourier filters. Visually comparing, the traditional filters show comparable performances and remove the noise to a considerable extent. The denoised images seem to have some degree of loss in structure when compared to the original images.

5.2 U-Net

Figure 7 shows the denoising results of U-Net model. The original (ground truth), noisy and denoised images allow to visually compare the outputs and understand U-Net's performance for the task of denoising. Looking at the denoised images, it can be seen that noise has been removed to a considerable extent. However, there seems to be significant amount of structural distortion in the denoised images resulting in minimal input features preserved.

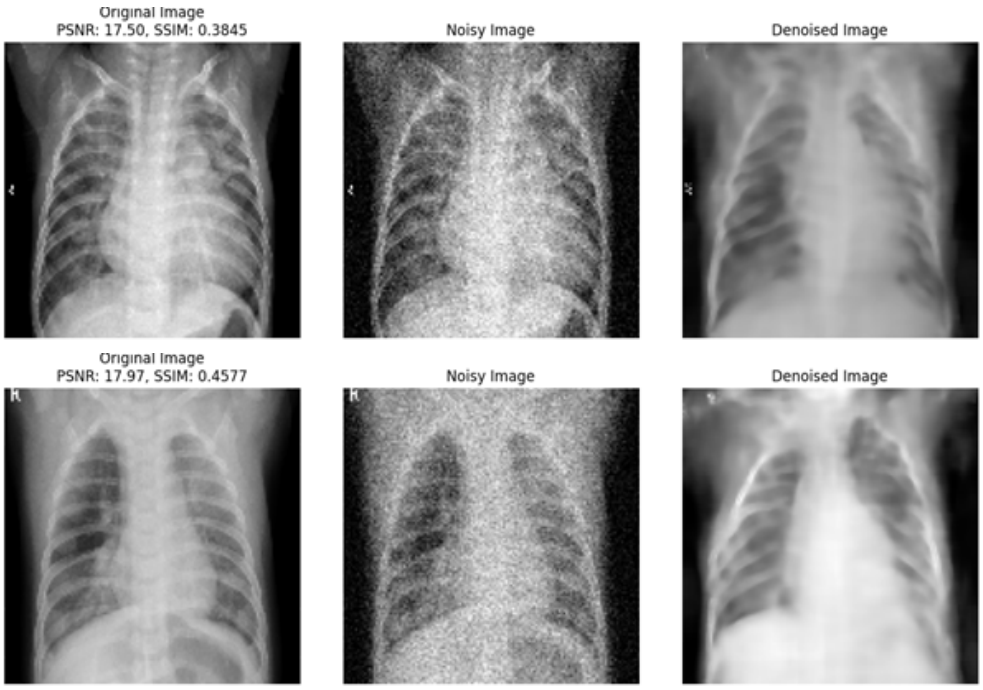

Figure 7: U-Net Denoising Results

5.3 GAN

Figure 8 depicts the noisy, clean (groud truth) and denoised set of images which summarize the denoising results of GAN model. Visually comparing the outputs, the denoised images output from GAN seem to contain certain degree of noise within them. The structural features of the input images are seen to be preserved in the denoised images to a considerable extent.

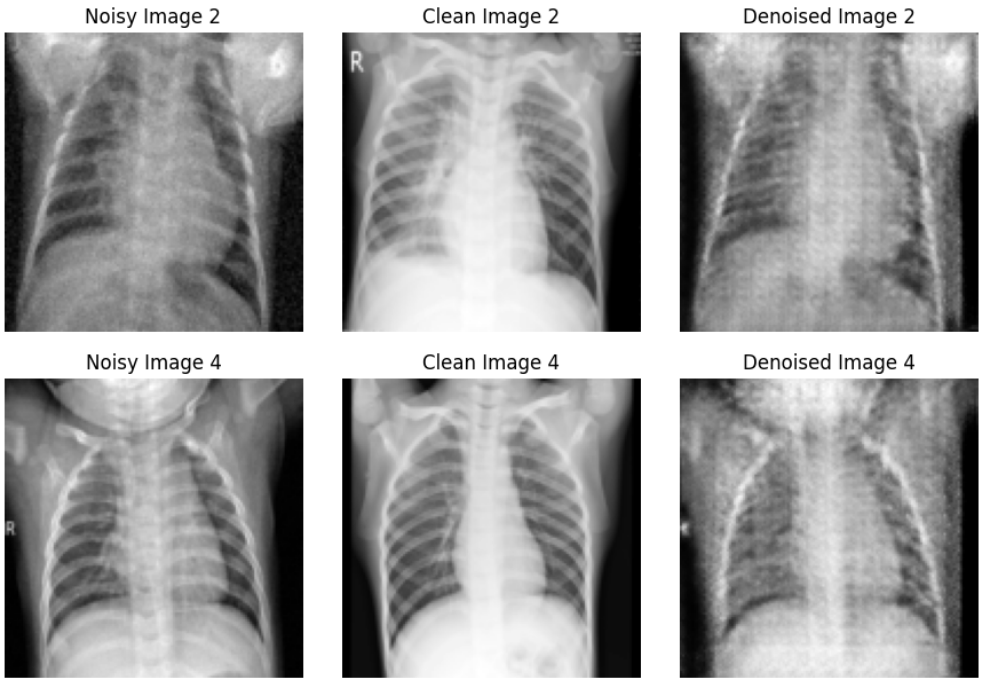

Figure 8: GAN Denoising Results

5.4 Autoencoder

Figure 10 shows the results of denoising for Autoencoder model. The denoised images show a significant amount of noise reduction for different types of noise along with a smoother appearance. The results also indicate that the structural features are preserved to a good extent post the denoising process.

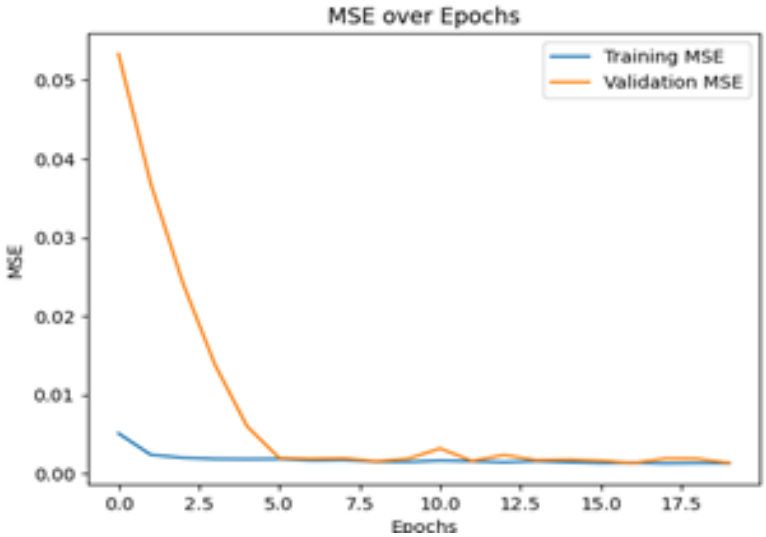

Figure 9: Training Loss curve

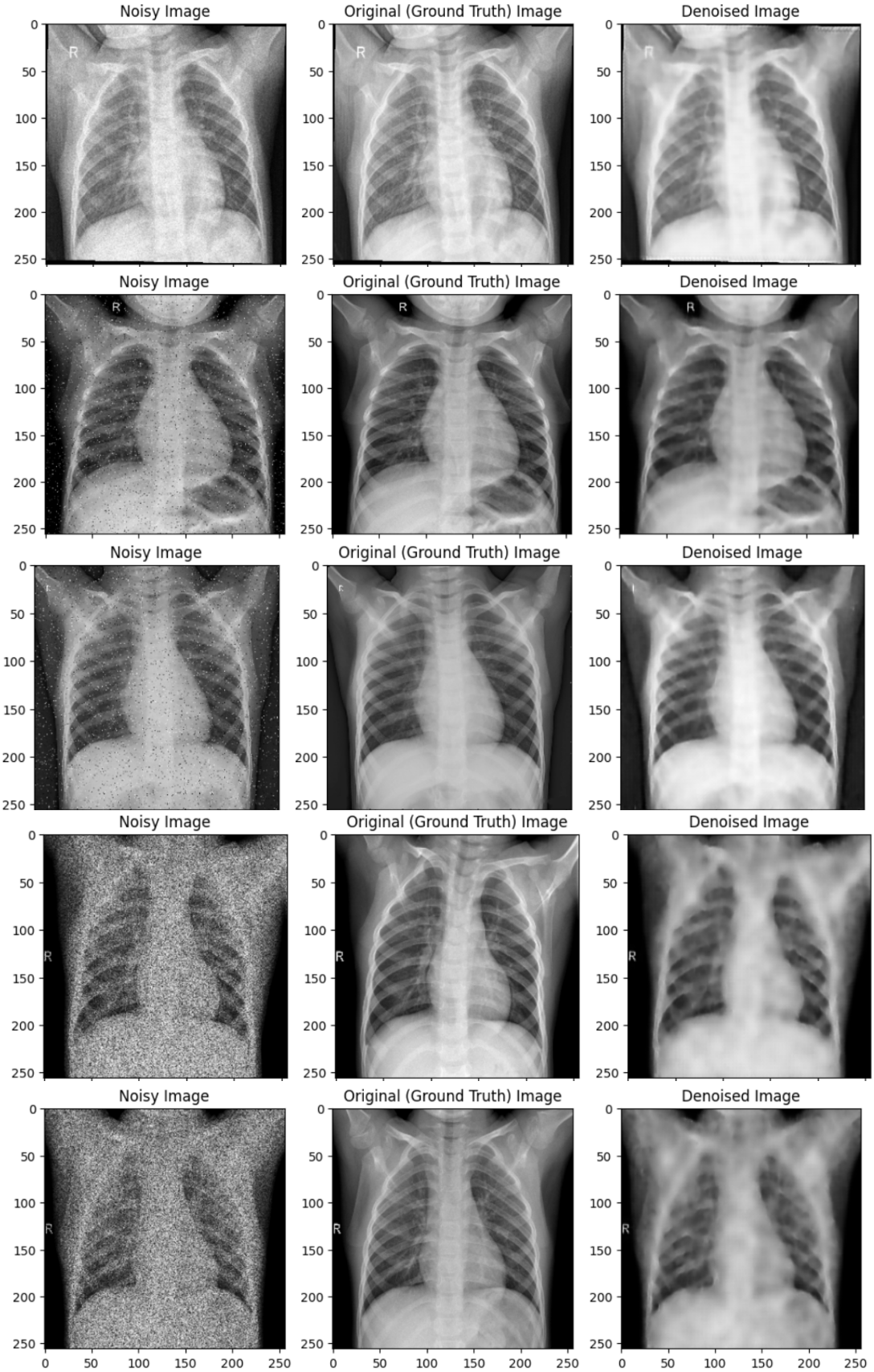

Figure 10: Autoencoder Denoising Results

- MSE vs Epoch graph: Figure 9 plots the Mean Squared Error (MSE) against epochs for the training process. As the number of epochs increases, MSE decreases. It shows our model is converging well and is gradually adjusting its parameters (weights) to minimize the error.
- Histogram Comparison: Figure 11 compares the histogram of all ground truth test images (original images) vs the histogram of all the predicted test images (denoised images). The histograms were plotted with number of bins=50. The ground truth histogram has a varied pixel distribution with a peak in pixel range: 0.6 to 0.7. The predicted image histogram is relatively more compressed with a peak in pixel range: 0.7-0.8. Visually comparing, it can be seen that there is significant similarity in the distribution of ground truth and denoised test images with some differences suggesting loss of fine details.

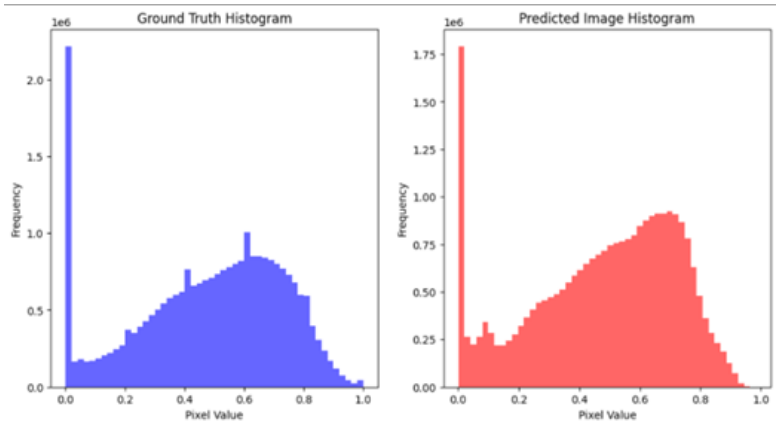

Figure 11: Histogram Comparison

6 Performance Evaluation

Table 1: Performance Evaluation of Traditional Filters and Deep Learning models

Denoising method	**Avg PSNR**	**Avg SSIM**
Mean Filter	28.13	0.44
Median Filter	28.16	0.49
Gaussian Filter	28.13	0.42
Bilateral Filter	28.13	0.43
Weiner Filter	28.13	0.35
Wavelet Filter	28.12	0.34
Fourier Filter	**28.12**	**0.57**
U-Net Model	15.7	0.37
GAN Model	14.21	0.3421
Autoencoder Model	**29.441** *	**0.801**

*When 24-bit depth images were used, a PSNR of 55 was obtained for the autoencoder model.

The traditional and deep learning methodologies were implemented on the testing dataset of 450 noisy images. The performance of all denoising techniques is measured using the below evaluation metrics[13]:

- **Peak Signal-to-Noise Ratio (PSNR)**: It computes the peak signal-to-noise ratio, in decibels, between two images. It is a ratio between the maximum possible power of an image and the power of corrupting noise that affects the quality of its representation. This ratio is used as a quality measurement between the original images (ground truth) and the denoised images. Typically, when the bit depth is 8 bits, values of PSNR are between 30 and 50 dB, where higher the value, better is the quality of the processed image.

  Formula:

  $$PSNR = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right)$$

  Where, MSE refers to the Mean Squared Error between the original and denoised image, MAX_I is the maximum possible pixel value

- **Structural Similarity Index Measure (SSIM):** It quantifies the image quality degradation caused by denoising process. It extracts 3 key features from an image: Luminance, Contrast and Structure. SSIM indicates the structural similarity between original (ground truth) and denoised images. The range of SSIM values is [0, 1] where values closer to 1 indicate higher structural similarity between original and processed images.
  Formula:

  $$SSIM(x, y) = [l(x, y)]^{\alpha} \cdot [c(x, y)]^{\beta} \cdot [s(x, y)]^{\gamma}$$

  where, $I(x, y)$ is the luminance comparison,
  $c(x, y)$ is the contrast comparison,
  $s(x, y)$ is the structure comparison

As shown in Table 1, average PSNR and SSIM values are computed for all 450 test images for each denoising technique. All the traditional models perform consistently in terms of PSNR (around 28). The Fourier Transform and Median Filter are superior in structural preservation (SSIM around 0.5).

With an average PSNR of 29.441[†] and an average SSIM of 0.801, **the Autoencoder performs best** amongst all the models. It significantly outperformed traditional methods and other deep learning approaches in terms of both PSNR and SSIM. The PSNR value is near 30, and the SSIM value is close to 1 given that the images were of 8 bits depth. This indicates a significant amount of noise removal, leading to better quality denoised images and higher levels of structural similarity between denoised and ground truth images.

U-net and GAN underperformed compared to traditional methods. However, they showcased adequate results considering the limited dataset and computational power. The performance of U-net and GAN can be improved with a larger training dataset, more computational resources, and hyperparameter tuning.

[†]When 24-bit depth images were used, a PSNR of 55 was obtained for the autoencoder model.

7 Conclusions

This project focused on enhancing Chest X-ray image quality degraded by noise from acquisition and transmission through the process of denoising. Chest X-ray pneumonia dataset was used. Different types of noises like, Gaussian, Impulse, Salt and Pepper, Poisson and Speckle noise were added to the images to curate a dataset of noisy and ground truth (clean) chest X-ray images. A variety of traditional and deep learning based denoising methods were implemented and their performances were assessed. The traditional filters implemented include Mean, Median, Gaussian, Bilateral, Wiener, Wavelet, and Fourier filters. The traditional filters were seen to be effective in some cases depending on the noise pattern, however, on complex noise patterns the resulting images contained certain degree of noise and structural distortion. Comparatively, the Fourier Transform and Median Filter showed superior structural preservation (SSIM around 0.5).

Deep learning based denoising techniques implemented include, U-Net, GAN and Autoencoder. Although U-net was able to efficiently preserve structural features of the input image and GAN, utilizing the adversarial learning was able to enhance the denoised image quality, both methods struggled with the relatively small dataset and limited computational resource constraints which affected the overall performance. Autoencoder came up to be the best performer judging by the highest PSNR (29.441) and SSIM (0.801). It balanced structural preservation and noise reduction well thanks to the ability of its latent representations to compress noisy images and reconstruct them with minimal distortion. Additionally, its relatively simple architecture and training process enhances its practical utility in medical imaging.

To some extent, the traditional noise cleaning methods were easy to understand and implement for cleaning basic noise. However, deep learning models provide real advantages for noisy scenarios beyond the basic level. Among these models, the Autoencoder performed best and had an adequate foundation for medical image quality improvement and accurate diagnostics. These findings serve as a starting point for future research involved in more correctly optimizing deep learning models for medical imaging.

8 Future Work

The results of this project show that both traditional and deep learning based methods can help to denoise chest X-ray images. Nevertheless, there are many avenues to improve the performance and overcome the limitations uncovered in this study. Increasing the size of the dataset can help to increase the generalization capabilities of the deep learning models. When models like U-Net or GAN are fed a larger-sized dataset, the features that they learn can become more resilient to different noise types and image variations. Experiments can be performed to fine-tune the hyperparameters of the deep learning models in order to optimize their learning and improve their denoising ability. Additionally, advanced architectures like diffusion models can be explored to obtain results with higher degrees of noise removal and structural preservation.

9 Personal Views and Reflections

9.1 Hossein Fathollahian

Computer vision is a field that links human perception and machine learning. This journey provided a solid understanding of its multi-level structure, from low-level to high-level techniques. I learned to understand and segment images at the medium level using clustering techniques and key point detection methods. In addition, RANSAC taught me how to manage noise and outliers. Explored key machine learning techniques in computer vision, including K-means clustering for image segmentation, Mean Shift for density-based clustering, and Regression for predictive analysis. These methods enhance the understanding of feature extraction and pattern recognition in visual data.

The skills and knowledge I gained have prepared me for professional settings. Working on diverse topics, I built a solid foundation for practical challenges. Collaborating with the team has enhanced my problem-solving abilities, critical thinking, and teamwork skills. These experiences have strengthened my confidence in applying computer vision to real-world applications.

Finally, high-level computer vision opened a world of possibilities with CNNs. The ability of neural networks to learn complex patterns is essential in solving visual tasks. This progression from fundamental concepts to advanced methods has given me a comprehensive understanding of the field.

9.2 Shivam Moudgil

Computer vision is an innovative technology that helps a machine identify patterns from pictures. Through this course, I have gained a good understanding of filters, **edge detection**, **Hough transforms**, segmentation, **Geometric Transformation**, feature detection, and the use of neural networks in visual recognition during this course. Of all these, I found the **Canny edge detector** most interesting because it notes delicate transitions in images, which makes it appropriate for object boundaries. Introduction to neural networks was also fascinating as it gave the potential to learn complex patterns in images, recognize objects and perform various classification tasks. This **Denoising project** builds upon the knowledge gained through this course. Apart from implementing traditional filtering methods, I explored how modern architectures like **Autoencoders** compress and reconstruct data, making them appropriate for reducing noise while retaining essential features. The Generative Adversarial Networks **(GANs)**, introduced me to an exciting species of learning known as adversarial learning, wherein the generator and the discriminator combine to create sharp, realistic outputs, which is a compelling approach toward fine detail restoration.

I wish to continue my learning in computer vision; I am interested in learning more about 3D Computer Vision and generative Models like GANs. I would like to explore more on GANs and Variational Autoencoders, to create realistic data, which can be applied to fields like medical imaging, art, and data augmentation.

9.3 Chirag Shinde

This course helped me establish a sharp understanding of computer vision and it main concepts like filtering images, edge detection and feature extraction which are important in the

interpretation of images. I especially liked reviewing such steps as SIFT and RANSAC which demonstrate how the semantic meaning is applied and how the robust feature matching is performed. Based on this course, I have learned the need to go further and understand more about real-time tracking, and deep learning in computer vision. Medical Imaging plays a huge role in diagnostics and is widely used for diagnosing various things, and I think that Computer Vision can be used to improve and better understand these results. Working on this project, particularly on the dataset and programming GAN to denoise medical images, was an enriching experience for me. This hands-on work allowed me to understand better how designing data pipelines advocates what the real world will be like, from noise augmentation to creating training and validation splits that mimic the data that we expect to have in real-world scenarios.

Just as enlightening as programming the GAN was. This gave me a chance to learn more about adversarial learning and how these generators and discriminators play together to generate outputs. Working on the GAN and debugging and optimizing it made me comfortable with hyperparameter tuning, loss functions, and the nuances of architectural issues such as upsampling and downsampling layers. I found it extremely satisfying to see the GAN getting better with further iterations, and it put in to perspective how iterative development can be so important for machine learning projects. Yet when I look back, there were things I could have worked on even better. Model training efficiency was one area where improvement was needed. Having strategies like employing pre trained models or more advanced optimizers could have fastened the convergence and altered results.Overall, this has been an amazing learning journey for myself. The experience gained will enable me to move forward building upon this to explore more advanced GAN architectures and involve in novel solutions in medical imaging.

9.4 Sushanth Gaurav Thota

This subject provided a strong foundation in computer vision, covering topics from basic image filtering and edge detection to advanced concepts like deep learning and neural networks. Early weeks introduced techniques such as **image filtering**, **Canny edge detection**, and **Hough transforms**, which helped me understand how to extract meaningful features from images. I also learned about **color segmentation**, **feature detection**, and advanced topics like **geometric transformations**, **RANSAC**, and **alignment**, all of which demonstrated how computer vision systems handle real-world data variability. For my project, I explored **image denoising** using both traditional and modern methods, learning about basic filters like Gaussian and median, as well as advanced techniques like convolutional autoencoders, **Generative Adversarial Networks (GANs)**, and **UNet**. These methods helped me understand the balance between noise reduction and detail preservation, and the role of adversarial training and skip connections in enhancing model performance.

While working on the project, I also researched industry-standard frameworks such as **MediaPipe**, **Movenet**, and **TensorFlow**, which are used to build robust applications. Although the team chose a different approach, I gained valuable insight into how diverse frameworks can simplify real-world application development by offering prebuilt tools for specific tasks. Overall, this subject bridged the gap between theory and practice, enhancing my technical skills and providing me with a broader understanding of tools and techniques relevant to the computer vision industry.

9.5 Mitravinda Manjunath

The field of computer vision (CV) has always intrigued me due to its ability to mimic human vision and its widespread applications in everyday life. This course has provided me with a comprehensive understanding of the complexity of CV and the foundational techniques that enable computers to process and interpret visual data. Starting from the basics, I gained a solid grasp of what constitutes an image, how filtering can enhance or modify images, and the significance of techniques like edge and corner detection. These foundational skills opened the door to understanding more advanced concepts, such as image segmentation and feature descriptors. Through this learning journey, I have come to appreciate the intricacies involved in some CV-based applications used in our daily lives, like face recognition and Adobe Photoshop.

Among the many topics covered, I enjoyed studying feature extraction and feature descriptors through techniques like SIFT and RANSAC. These methods provided fascinating insights into how semantic meaning is derived from images and how robust matches are established between features in different images. I see these techniques as crucial building blocks for preparing high-quality image datasets that power state-of-the-art deep learning and image generation models. Reflecting on the course, I feel inspired to delve deeper into advanced topics such as real-time tracking, CV for video datasets and especially CV techniques involving deep learning architectures as they represent the frontier of computer vision innovation. This course and this project has not only enhanced my technical understanding but also fueled my curiosity to explore the transformative potential of computer vision in solving real-world problems.

10 Contribution

10.1 Hossein Fathollahian

- **Literature Review and Learning** In the first part, I worked on different traditional method literature related to many years ago and compared different methods for denoising[1]. The materials included in this review also described conventional noise reduction techniques and contemporary deep learning algorithms such as U-Net[4].
- **Implementing Traditional models** I worked on implementing various traditional models like the Median filter, Weiner filter, etc, for denoising the image.
- **Training of the U-Net Model** I was also involved in the training phase of the Unet model. We went through various existing architectures of Unet to understand it and then built upon it.[5].
- **Testing and Evaluation** I tested the model, employing conventional and deep learning approaches on noisy chest X-ray datasets, which let me judge the efficacy of the U-Net model against simple filters[6].

10.2 Shivam Moudgil

- **Adding Noise to Dataset:** I was involved in task of adding various kinds of noise to the existing clean image dataset.
- **Literature Review For Autoencoder and GAN:** Before actual implementation, I

went through various existing materials on GAN and Autoencoder architecture related to denoising images.[12] [9]

- **Formulating Autoencoder Architecture** I was involved in creating autoencoder architecture for our system. I took some guidance from existing code on typical autoencoder architecture[3] and then built our model by adding significant modifications to Convolutional and bottleneck layers.
- **Performance Evaluation**: I researched about various performance metrics which can be used to evaluate the denoised images[13]. I evaluated performance metrics like average PSNR and SSIM for all the resultant denoised images for various models.

10.3 Sushanth Gaurav Thota

- **Literature Review for Autoencoder:** Before the implementation phase, I conducted an in-depth review of existing materials on Autoencoder architecture for image denoising[12].
- **Training the Autoencoder Model:** I implemented and trained the Autoencoder model by adding convolutional layers, ReLU activations, and batch normalization layers. Additionally, I integrated transpose convolutional layers in the decoder for upsampling the image. I utilized Adam optimizer and tuned the learning rate and batch size for optimal performance during training[3].
- **Hyperparameter Tuning** I fine-tuned hyperparameters such as the number of convolutional filters, kernel size, learning rate, and batch size to improve the model's denoising accuracy.
- **Testing**: After training, I tested the model on a separate test set of noisy-clean image pairs. The noisy images were fed into the model to generate denoised outputs, allowing me to assess its ability to generalize and restore image quality.[13].

10.4 Chirag Deepak Shinde

- **Literature Review for GAN and U-Net:** I performed a comprehensive assessment of previous studies regarding applying GAN and U-Net architectures to image denoising. This allowed me to learn their working methods, features, and drawbacks Before constructing the GAN model for this project[4].
- **Data Preparation for GAN:** I preprocessed the dataset of chest X-ray images by normalizing and standardizing the sizes of the images to the recommended GAN format. To make it easier to train and test the model, clean-noisy image pairs were arranged as follows.
- **Formulating GAN Architecture:** I defined the GAN architecture by developing a generator for reconstructing noisy images to clean ones and discriminator for identifying fake and real images. The architecture incorporated adversarial learning, and the best results were obtained when the system was trained for denoising chest X-ray images. I also took some assistance from[14] for better understanding of GAN architecture.

- **Testing the GAN Model:** I also applied the trained GAN model on a different test set to test for its efficiency in removing noise without eradicating structure information. I validated model's outputs with the clean dataset to confirm correct removal of noise and retention of critical structural details during validation.

10.5 Mitravinda Manjunath

- **Literature Survey of Traditional filters, GAN and Autoencoder:** I performed extensive literature review of different traditional and deep learning methodologies used for denoising medical images including survey papers[1;2] and architecture specific papers[9;8].
- **Training GAN model:** I implemented the GAN architecture and trained the Generator and Discriminator networks of the model. I executed convolutional and batch normalization layers with ReLU activation function during downsampling; transposed convolutional and batch normalization layers with ReLU activation function during upsampling.
- **Hyperparameter tuning:** I performed multiple rounds of experiments to tune the hyperparameters- epochs, batch size, learning rate, kernel size and stride, so that the performance of GAN is maximized.
- **Performance evaluation and deriving insights:** I researched on suitable performance metrics to evaluate the denoising process. I performed testing and evaluation using PSNR and SSIM metrics on the trained GAN model. Further, I took part in tabulating the results of all denoising methods; derived insights and inferences on the overall task of image denoising.

References

[1] Subhagata Chattopadhyay. A study on various common denoising methods on chest x-ray images. *Artificial Intelligence Evolution*, pages 87–106, 2022.

[2] Walid El-Shafai, Samy Abd El-Nabi, Anas M Ali, El-Sayed M El-Rabaie, and Fathi E Abd El-Samie. Traditional and deep-learning-based denoising methods for medical images. *Multimedia Tools and Applications*, 83(17):52061–52088, 2024.

[3] Garima. Denoising autoencoder. https://github.com/Garima13a/Denoising-Autoencoder.

[4] Mattias P Heinrich, Maik Stille, and Thorsten M Buzug. Residual u-net convolutional neural network architecture for low-dose ct denoising. *Current Directions in Biomedical Engineering*, 4(1):297–300, 2018.

[5] Jack. Unet-based image denoising. https://github.com/ijackyang/Unet-Image-Denoise, 2024.

[6] Yan Jin, Xiao-Ben Jiang, Zhen-kun Wei, and Yuan Li. Chest x-ray image denoising method based on deep convolution neural network. *IET Image Processing*, 13(11):1970–1978, 2019.

[7] Daniel Kermany. Labeled optical coherence tomography (oct) and chest x-ray images for classification. *Mendeley data*, 2018.

[8] Donghoon Lee, Sunghoon Choi, and Hee-Joung Kim. Performance evaluation of image denoising developed using convolutional denoising autoencoders in chest radiography. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 884:97–104, 2018.

[9] Yuqin Li, Ke Zhang, Weili Shi, Yu Miao, and Zhengang Jiang. A novel medical image denoising method based on conditional generative adversarial network. *Computational and Mathematical Methods in Medicine*, 2021(1):9974017, 2021.

[10] Nahida Nazir, Abid Sarwar, and Baljit Singh Saini. Recent developments in denoising medical images using deep learning: An overview of models, techniques, and challenges. *Micron*, 180:103615, 2024.

[11] Dang NH Thanh, P Kalavathi, VB Surya Prasath, et al. Chest x-ray image denoising using nesterov optimization method with total variation regularization. *Procedia Computer Science*, 171:1961–1969, 2020.

[12] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 1096–1103. ACM, 2008.

[13] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[14] Jakub Zdanowski. Gan-based image denoising. https://github.com/INFJakZda/GAN-Image-Denoising, 2024.