PRESIDENCY UNIVERSITY

Private University Estd. in Karnataka State by Act No. 41 of 2013

MEDICAL IMAGE DENOISING USING AUTOENCODERS, CNN & DEEP LEARNING

A Project Report Submitted in Partial

Fulfilment of the Requirements

for the Degree of

BACHELOR OF TECHNOLOGY

in

Computer Science and Engineering

By

TEAM NUMBER-52

Mohammed Abdullah – 20181CSE0426

Mohammed Saif – 20181CSE0433

Mohammed Uvais – 20181CSE0437

Nishithaa Palani – 20181CSE0490

Syed Yunus – 20181CSE0737

GUIDE: Mr. Yamanappa-Asst.Prof-CSE

to

SCHOOL OF ENGINEERING

PRESIDENCY UNIVERSITY

BANGALORE – 560 089

MAY 2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF ENGINEERING

PRESIDENCY UNIVERSITY

CERTIFICATE

This is to certify that the project report **"MEDICAL IMAGE DENOISING USING AUTOENCODERS, CNN & DEEP LEARNING"** being submitted by *"Mohammed Abdullah, Mohammed Saif, Mohammed Uvais, Nishithaa Palani and Syed Yunus"* bearing roll number(s) *"20181CSE0426, 20181CSE0433, 20181CSE0437, 20181CSE0490, 20181CSE0737"* respectively in partial fulfilment of requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a bonafide work carried out under my supervision.

10/06/22

Dr. C. KALAIARASAN
Associate Dean – Admin
Department of CSE
Presidency University

10/06/22

Mr. Yamanappa
Assistant Professor
Department of CSE
Presidency University

DECLARATION

We hereby declare that the report entitled **"Medical Image Denoising Using Autoencoders, CNN & Deep Learning"** which is being submitted to the Presidency University, Bangalore is a bonafide report of the work conducted by us. The material contained in this report is not submitted to any University or Institution for the award of any degree.

Name	ID	Department	Signature
Mohammed Abdullah	20181CSE0426	CSE	
Mohammed Saif	20181CSE0433	CSE	
Mohammed Uvais	20181CSE0437	CSE	
Nishithaa Palani	20181CSE0490	CSE	
Syed Yunus	20181CSE0737	CSE	

ACKNOWLEDGEMENT

We want to extend a sincere and heartfelt obligation towards all the personages without whom the completion of the project was not possible. We express our profound gratitude and deep regard to Mr. Yamanappa [Assistant Professor CSE], Presidency University Bangalore for his guidance, valuable feedback, and constant encouragement throughout the project. His valuable suggestions were of immense help. We sincerely acknowledge his constant support and guidance during the project.

We are immensely grateful to all the team members for their constant support and encouragement. We are also grateful to the Presidency University, Bangalore, for allowing us to do this project and providing all the required facilities.

Bangalore – 560 089

[May 2022]

Mohammed Abdullah

Mohammed Saif

Mohammed Uvais

Nishithaa Palani

Syed Yunus

ABSTRACT

Image denoising is an important pre-processing step in medical image analysis. The basic intent of image denoising is to reconstruct the original image from its noisy observation as accurately as possible, while preserving important detail features such as edges and textures in the denoised image. Different algorithms have proposed in the past three decades with varying denoising performances. The sources of noise present significant problems to image denoising. Gaussian, impulse, salt, pepper, and speckle noise, in particular, are complex sources of noise in imaging. In the task of image denoising, the convolutional neural network methodology gained a lot of attention in recent years. Several CNN approaches for image denoising have been proposed so far. These models work well only for certain noise models and specific to datasets. Here we are developing a CNN model to work for different noise models and noise levels in medical imaging modalities like MRI, Ultrasound, OCT (Optical Coherent Tomographic) images.

Our main objective is to remove the noise while preserving the vital details like edges, texture which are more important for any radiologist to diagnose abnormality in anatomical images.

KEYWORDS: Image Processing, Image Denoising, CNN, MRI, Ultrasound, OCT images, PSNR, SSIM Types of Noise, Spatial domain filtering, Variational denoising methods, Transform Techniques in Image Denoising, BM3D, Encoders and Decoders, Extensive Empirical Study, Medical Image Analysis, Gaussian Noise, Impulse Noise, Salt Noise, Pepper Noise, Speckle Noise, Data sets of COCO, Kaggle, MIT-Adobe, etc.

CONTENTS

1. INTRODUCTION

Digital images are used in everyday applications such as magnetic resonance imaging, ultrasonic imaging, and optical computed tomography, as well as in research and technology fields such as geographic information systems and astronomy. In the realm of image processing, noise removal is one of the most crucial aspects. During the transmission and reception processes, an image is distorted by many sorts of noise. Substitute noise, speckle noise, and additive white Gaussian noise are the diverse types of noise.

As a result, denoising medical images is even more important, allowing physicians to do more exact illness analysis. X-RAY, OCT (Optical Computed Tomography), MRI (Magnetic Resonance Imaging), PET (Positron Emission Tomography), and SPECT (Single Photon Emission Computed Tomography) are medical scans that contain minute details about the heart, brain, and nerves. X-ray Optical Computed Tomography (OCT) is a strong technology for detecting an object's interior structure. As a result, it defines applications, such as non-destructive testing of a wide range of materials. The OCT image is obtained from a large number of systematic observations at various viewing angles, and the final OCT image is then reconstructed using a computer (Radon transform). When medical images are contaminated by noise, it is impossible to save a human being from injury. OCT image denoising is a major research topic in both Image Processing and Biomedical Engineering. In the case of OCT, a variety of mathematical applications can be used to determine whether the normal tissue has been infected by the cancer cell's mutations. Denoising the OCT pictures, which removes noise, has improved the efficiency of the illness diagnosis technique. The denoised photos show a significant increase in PSNR values, resulting in a smoother image for diagnosing purposes. A number of approaches have been developed to improve the quality of OCT pictures. While various methods have been developed for image denoising, the problem of image noise suppression remains unsolved, especially in instances when the photographs were taken in low lighting with a high degree of noise. We give a wide evaluation of medical image denoising in the spatial domain and transform domain in this paper, with each having its own assumptions, limitations, and advantages [1].

1.1 Definition of Image Denoising

Noise is nothing but the unwanted spurious information in the digital images is called Noise. Noise is a random variation of the image density that result in Pixel values that do not reflect the true intensities of the real scene. Image denoising is a process with which we reconstruct original image from a Noisy one. Sources of noise in an image mostly occur during storage, transmission and acquisition of the image [2]. Image denoising is to remove noise from a noisy image, to restore the true image. However, since noise, edge, and texture are high frequency components, it is difficult to distinguish them in the process of denoising and the denoised images could inevitably lose some details [3].

1.2 Why Image Denoising is Important?

Image Denoising plays a significant role in a wide range of applications such as **Image Restoration, Visual Tracking, Image Registration, Image Segmentation, and Image Classification**, where obtaining the original image content is crucial for robust performance. The purpose of denoising is to remove the noise while retaining the edges and other detailed features as much as Possible. Denoising suppresses the adverse impact of noise in the acquired image. It is an important pre-processing step to achieve effectiveness and robustness with many image processing algorithms like segmentation, feature extraction, texture analysis etc. While many algorithms have been proposed for the purpose of image denoising, the problem of image noise suppression remains an open challenge, especially in situations where the images are acquired under poor conditions where the noise level is very high [4].

1.3 What is Neural Networks?

Artificial neural networks (ANNs) and simulated neural networks (SNNs) are subsets of machine learning that are at the heart of deep learning methods. Their name and structure are based on the human brain, and they function similarly to organic neurons.

A node layer consists of an input layer, one or more hidden layers, and an output layer in artificial neural networks (ANNs). Each node, or artificial neuron, is connected to the next and has a weight and threshold associated with it. If a node's output exceeds a certain threshold, the node is activated, and data

is sent to the network's next tier. Aside from that, no data is continued to the network's next tier [5].

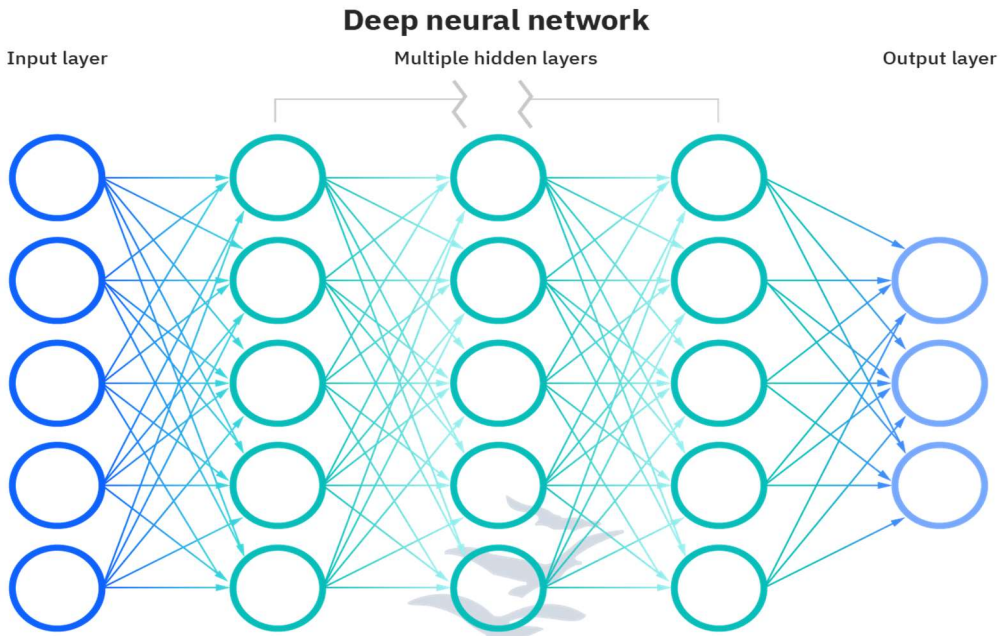


Fig 1. Deep Neural Network Schema

1.4 Types of Neural Networks

Particular types of neural networks exist, each of which serves a different purpose. While this isn't an exhaustive list, the following are some of the most popular types of neural networks that you'll come across for common applications [6]:

1.4.1 Artificial Neural Network (ANN):

A set of several perceptron's or neurons at each layer makes up Artificial Neural Network (ANN). Because inputs are only processed in one way, an ANN is also known as a Feed-Forward Neural Network. One of the most basic types of neural networks is this type. They transmit data in one direction, passing it through multiple input nodes until it reaches the output node. Hidden node layers in the network may or may not exist, making its operation more understandable.

✞ **Advantages:**

- Storing information on the entire network.

- Ability to work with incomplete knowledge.
- Having fault tolerance.
- Having a distributed memory. ✞ **Disadvantages:**
- Hardware dependence.
- Unexplained behavior of the network.
- Determination of proper network structure.

1.4.2 Recurrent Neural Network (RNN):

RNNs are more complicated than recurrent neural networks. They record the output of processing nodes and input it back into the model (they did not pass the information in one direction only). The model is said to learn to anticipate the outcome of a layer in this manner. Each node in the RNN architecture serves as a memory cell, allowing calculation and operation to continue. During backpropagation, if the network's forecast is inaccurate, the system self-learns and works towards the correct prediction.

✞ **Advantages:**

- An RNN remembers each and every information through time. It is useful in time series prediction only because of the feature to remember previous inputs as well. This is called Long Short-Term Memory.
- Recurrent neural network is even used with convolutional layers to extend the effective pixel neighborhood.

✞ **Disadvantages:**

- Gradient vanishing and exploding problems.
- Training an RNN is an exceedingly challenging task.
- It cannot process exceptionally long sequences if using tanh or relu as an activation function.

	ANN	*CNN*	*RNN*
Type of Data	Tabular Data, Text Data	Image Data	Sequence data
Parameter Sharing	No	Yes	Yes
Fixed Length input	Yes	Yes	No
Recurrent Connections	No	No	Yes
Vanishing and Exploding Gradient	Yes	Yes	Yes
Spatial Relationship	No	Yes	No
Performance	ANN is considered to be less powerful than CNN, RNN.	CNN is considered to be more powerful than ANN, RNN.	RNN includes less feature compatibility when compared to CNN.
Application	Facial recognition and Computer vision.	Facial recognition, text digitization and Natural language processing.	Text-to-speech conversions.
Main advantages	Having fault tolerance, Ability to work with incomplete knowledge.	High accuracy in image recognition problems, Weight sharing.	Remembers each and every information, Time series prediction.
Disadvantages	Hardware dependence, Unexplained behavior of the network.	Large training data needed, don't encode the position and orientation of object.	Gradient vanishing, exploding gradient.

Fig 2. Comparison between ANN, CNN & RNN

1.4.3 Convolutional Neural Network (CNN):

A convolutional neural network (CNN) is a form of artificial neural network that is specifically intended to process pixel input and is used in image recognition and processing.

CNNs are powerful image processing, artificial intelligence (AI) that use deep learning to perform both generative and descriptive tasks, often using machine vison that includes image and video recognition, along with recommender systems and natural language processing.

✞ **Advantages:**

- Very High accuracy in image recognition problems.
- Automatically detects the prominent features without any human supervision.
- Weight sharing.

✞ **Disadvantages:**

- CNN do not encode the position and orientation of object.
- Lack of ability to be spatially invariant to the input data.
- Lots of training data is required

1.5 What is CNN?

The convolutional neural network (CNN), sometimes known as CNN, is a type of deep learning neural network. Concisely, consider CNN to be a machine learning system that can take an input image, assign relevance (learnable weights and biases) to various aspects/objects in the image, and distinguish between them.

CNN works by extracting features from the images. Any CNN consists of the following:

- The input layer which is a grayscale image
- The Output layer which is a binary or multi-class labels
- Hidden layers consisting of convolution layers, ReLU (Rectified Linear Unit) layers, the pooling layers, and a fully connected Neural Network

It is especially important to understand that ANN or Artificial Neural Networks, made up of multiple neurons is not capable of extracting features from the image. This is where a combination of convolution and pooling layers comes into the picture. Similarly, the convolution and pooling layers can't perform classification hence we need a fully connected Neural Network [7].

CNNs are powerful image processing, artificial intelligence (AI) that use deep learning to perform both generative and descriptive tasks, often using machine vison that includes image and video recognition, along with recommender systems and natural language processing

A neural network is a system of hardware and/or software patterned after the operation of neurons in the human brain. Traditional neural networks are not ideal for image processing and must be fed images in reduced-resolution pieces. CNN have their "neurons" arranged more like those of the frontal lobe, the area responsible for processing visual stimuli in humans and other animals. The layers of neurons are arranged in such a way as to cover the entire visual field avoiding the piecemeal image processing problem of traditional neural networks.

A CNN employs a system similar to a multilayer perceptron that is optimized for speed. An input layer, an output layer, and a hidden layer that contains numerous convolutional layers, pooling layers, fully connected layers, and normalizing layers make up a CNN's layers. The removal of constraints and increase in image processing efficiency leads in a system that is significantly

more effective and easy to train for image processing and natural language processing [8].

1.6 Activation Functions

To put it another way, an artificial neuron calculates the 'weighted total' of its inputs and then adds a bias, as represented in the diagram below by the net input [9].

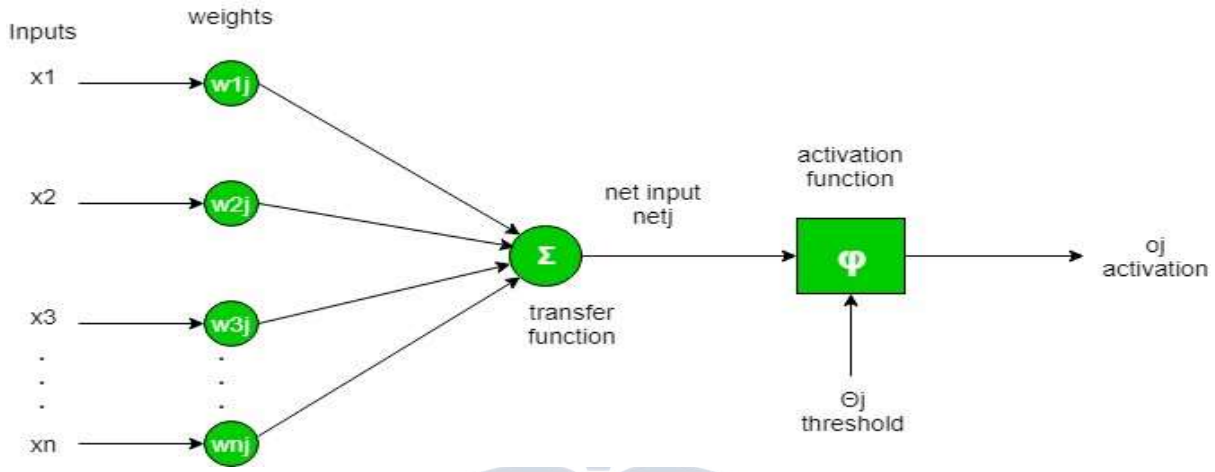


Fig 3. Activation Functions

1.6.1 Types of Activation Functions

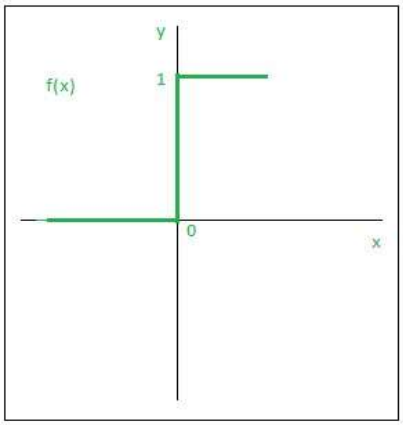
Several distinct types of activation functions are used in Deep Learning. Some of them are explained below [10]:

a) Step Function:

One of the most basic types of activation functions is the step function. We consider a threshold value, and the neuron is triggered if the value of net input, say y, is greater than the threshold.
Mathematically,

$$f(x) = 1, \text{if x} >= 0$$
$$f(x) = 0, \text{if x} < 0$$

Given below is the graphical representation of step function.

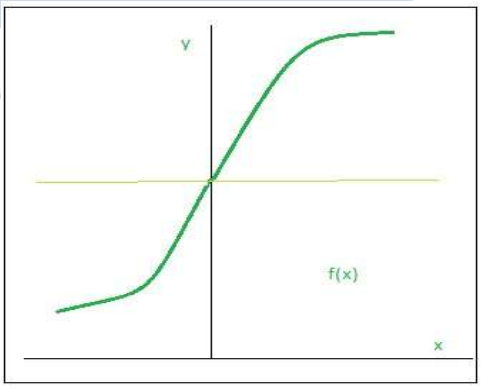


b) Sigmoid Function:

Sigmoid function is a widely used activation function. It is defined as:

$$\frac{1}{(1+e^{-x})}$$

Graphically,



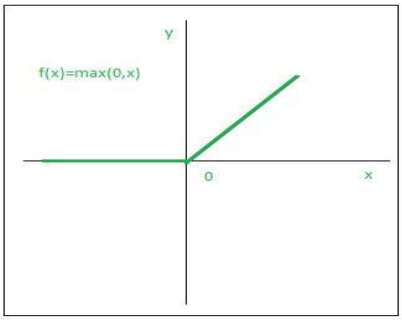
This is a smooth and continuously differentiable function. The fact that it is non-linear gives it an edge overstep and linear functions. The sigmoid function has a really cool property. This basically indicates that if I have several neurons with sigmoid activation functions, the output will be nonlinear as well. The function has a S form and runs from 0 to 1.

c) ReLU:

The ReLU function is the Rectified linear unit. It is the most widely used activation function. It is defined as:

$$f(x) = \max(0, x)$$

Graphically,



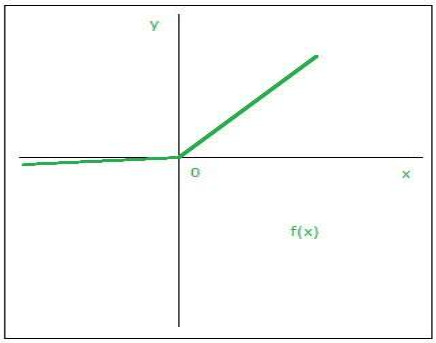
The key benefit of employing the ReLU function over other activation functions is that it does not simultaneously stimulate all of the neurons. What exactly does this imply? If the input is negative, the ReLU function will convert it to zero, and the neuron will not be engaged.

d) Leaky ReLU:

The Leaky ReLU function is a better variant of the ReLU function. We define the Relu function as a small linear component of x, rather than specifying it as zero for x less than 0. It is defined as follows:

$$f(x) = ax, x < 0$$
$$f(x) = x, \text{otherwise}$$

Graphically,



1.7 Convolutional Layers

Convolutional layers are useful for extracting features from images because they deal with spatial redundancy by sharing weights. The characteristics get more exclusive and informative as we progress deeper into the network, and redundancy is decreased. This is mostly due to repeated cascaded convolutions and subsampling layer information compression. We end up with a compressed feature representation of the image's content as redundancy is eliminated [11].

1.8 CNN Pooling Layer

1.8.1 Max Pooling

Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter. Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map.

1.8.2 Average Pooling

Average pooling computes the average of the elements present in the region of feature map covered by the filter. Thus, while max pooling gives the most prominent feature in a particular patch of the feature map, average pooling gives the average of features present in a patch.

1.8.3 Global Pooling

Global pooling reduces each channel in the feature map to a single value. Thus, a $\mathbf{n_h \times n_w \times n_c}$ feature map is reduced to $\mathbf{1 \times 1 \times n_c}$ feature map. This is equivalent to using a filter of dimensions $\mathbf{n_h \times n_w}$ i.e., the dimensions of the feature map.

1.9 Types of Image Noise Models

There are several types of image noise. Let's Discuss some of them [12]. **1.9.1**

Gaussian Noise:

Gaussian noise is statistical noise having a probability density function (PDF) equal to the normal distribution, as is well known. The distribution of Gaussian noise is uniform throughout the signal.

In a noisy image, each pixel is composed of the sum of its original pixel values plus a random Gaussian noise value. A Gaussian distribution's

probability distribution function has a bell shape. The most typical application for Gaussian noise in applications is additive white Gaussian noise.

The below figure shows the Gaussian distribution function (probability distribution function) of Gaussian noise and pixel representation of Gaussian noise.

The probability density function p of a Gaussian random variable z is given by:

$$p_G(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z-\mu)^2}{2\sigma^2}}$$

where z represents the grey level, μ the mean grey value and σ its standard deviation.

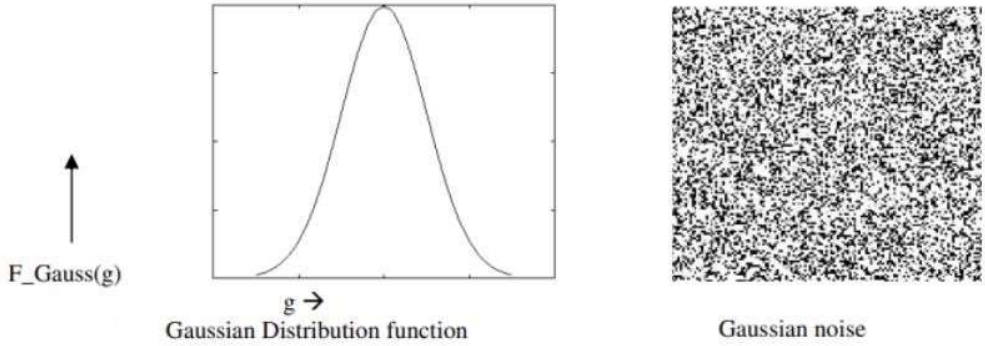


Fig 4. Gaussian Noise

1.9.2 Salt and Pepper Noise:

Salt and pepper noise is a sort of noise that is typically visible in pictures. It takes the form of white and black pixels that occur at irregular intervals. This type of noise is caused by data transport errors. In salt pepper noise, the values a and b are different. On average, each has a probability of less than 0.1. The image has a "salt and pepper" appearance because the corrupted pixels are alternately assigned to the lowest and highest value. This noise's distribution and pixel representation are presented below.

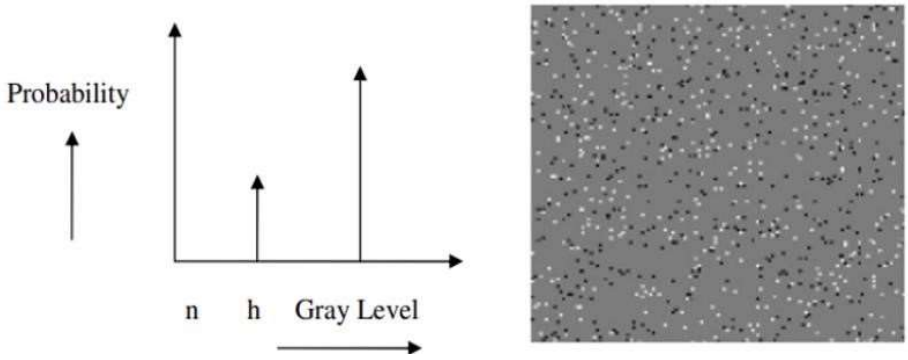


Fig 5. Salt and Pepper Noise

1.9.3 Speckle Noise:

Speckle noise, unlike Gaussian or Salt & Pepper noise, is multiplicative noise. This decreases image quality in diagnostic investigations by giving images a backscattered wave looks created by a large number of microscopic, dispersed reflections moving through internal organs. The observer will have a harder time distinguishing tiny features in the photographs as a result of this. This noise's distribution and pixel representation are presented below.

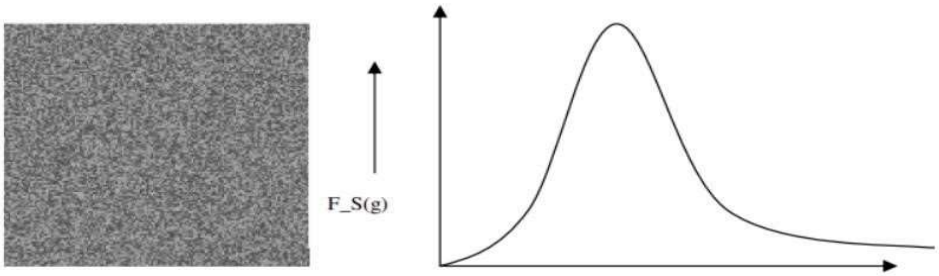


Fig 6. Speckle Noise

1.9.4 Poisson Noise:

The image detectors and recorders' nonlinear responses generate Poisson noise. The image data dictates this form of noise. Because arbitrary electron emission with a Poisson distribution and a mean response value is used in detecting and recording methods, this equation is used. The image-dependent term is assumed to have a standard deviation if the noise has a variance of one, because the mean and variance of a Poisson distribution are the same.
]

1.9.5 Impulse Noise:

In the discrete world, an impulse function on a value of 1 at a single place is an idealized function, while in the continuous world, an impulse function with unit area is an idealized function.

Fig 7. Impulse function in discrete world and continuous world

2. LITERATURE SURVEY

2.1 MRI

- **A Convolutional Neural Network for Denoising of Magnetic Resonance Images – 2020(PATREC 7847)**

They have proposed a new CNN model, namely CNN-DMRI, for reduction of Rician noise from MRI images. The proposed CNN consists of multiple convolutions which capture different image features while separating inherent noise. In addition, the proposed method performs down-sampling and up sampling of images in the denoising process through the encoder-decoder framework. The training of the network is conducted with synthetic MRI images.

From the implementation results, the proposed model attained PSNR of 43.18 and SSIM of 0.98.

- **MRI denoising using progressively distribution-based neural network – 2020(MRI 9412)**

In this work, a progressive learning strategy was successfully applied to MR image Rician denoising, via fitting the distribution at pixel-level and feature level. The progressive network called Rician Net with two cascaded sub-Rician Nets was proposed for achieving a better performance. Applying the BN layer, Conv layer and residual unit has improved the network performance. Moreover, verified that the wide network with large convolutional filters can obtain a larger acceptance domain, such that CNN can extract more available pixel distribution characteristics.

From the implementation results, the proposed model attained PSNR of 35.21 and SSIM of 0.96.

- **Optimal Bilateral Filter and Convolutional Neural Network based Denoising Method of Medical Image Measurements – 2019(MEASUR 6587)**

In this paper, an innovative bioinspired optimization-based filtering technique considered for the MI denoising process, the filtered named as BF. The choice of choosing optimal parameter is that is Gaussian and spatial weights affected the performance of the denoising process, these parameters selected by using swarm-based optimization that is DF and MFF algorithm. For parameter selection, multi-objective fitness (PSNR) is used. After the determination of optimal parameters, investigated the results of proposed BF parameters with medical images with the denoised process. Moreover, finally, CNN is used to classify the denoised image as normal or abnormal, with better classification rate.

From the implementation results, the proposed model attained PSNR of 52.35 and SSIM of 0.95.

2.2 ULTRASOUND

- **Ultrasound image DE-speckling by a hybrid deep network with transferred filtering and structural prior – 2020**

It is infeasible to acquire clean US images as the training data for de-speckling. To this end, they propose a hybrid network for reduction of US image speckle noise. Considering the problem of lack of labelled images, we have trained a transferable network depending on the Gaussian distribution prior to real US images. In addition, two VGGNets are used to extract structural boundaries from real US images for fine-tuning the pre-trained network. The proposed framework is applied to simulated US images and real US images and compared with five traditional methods and three CNN-based deep neural network methods.

From the implementation results, the proposed model attained PSNR of 30.68 and SSIM of 0.90.

- **DE-speckling of clinical ultrasound images using deep residual learning – 2020**

This paper proposes a novel method for DE-speckling using pre-trained residual learning network (RLN). Initially, RLN is trained with pristine and its corresponding noisy images in order to achieve a better performance. The developed method chooses a pre-trained RLN for DE-speckling with less computational resources. But the training procedure of RLN from scratch is computationally demanding. The pre-trained RLN is a blind DE-speckling approach and does not require any fine tuning and noise level estimation. The presented approach shows superiority in the removal of speckle noise as compared to the existing state-of-art methods.

From the implementation results, the proposed model attained PSNR of 31.46 and SSIM of 0.89.

2.3 IMAGE DENOISING TECHNIQUES

- **Wavelet enabled convolutional autoencoder based deep neural network for hyperspectral image denoising – 2021**

This article presents a dual branch DL based denoising method called WaCAEN to remove Gaussian, stripe, and mixed noise types effectively from HSI. To detect directional stripe noise as well as randomly distributed Gaussian noise, HDWT is perform on noisy bands that fetches directional as well as average properties of an image. This also allows the network to be efficient in terms of processing time without losing the denoising performance. The proposed method includes benefits of deep CNN, autoencoder, skip connections and subpixel up sampling to get the desired result. Experimental results demonstrate the superior performance of the proposed method compared to other tested methods on synthetic as well as real datasets. In addition to denoising of spatial dimension images, the proposed method also demonstrates its effectiveness in restoration of spectral signature which is a crucial property of HSI. Unlike the existing denoising methods, the proposed WaCAEN method efficiently oversees multiple types of noise that are frequently observed in real HSI.

From the implementation results, the proposed model attained PSNR of 32.68 and SSIM of 0.90.

- **Methods for image denoising using convolutional neural network: a review – 2021**

This paper mainly focuses on CNN architectures which are becoming quite useful in image denoising. They have proposed a survey of different techniques relating to CNN image denoising. A clear understanding of different concepts and methods was elucidated to give readers a grasp of contemporary trends. Several techniques for CNN denoising have been enumerated. A total of 144 references were included in this paper. From the study, we observed that the GAN was the most used method for CNN image denoising. Several methods used the generator and the discriminator for extraction and clean image generation. Interestingly, some researchers combined the GAN method with the DCNN methods. The feedforward CNN and U-Net were also used. Researchers severally used the residual network. A reason for the high usage of the residual network could be its effectiveness and efficiency. Researchers used the residual network to limit the numbers of convolutions in their network.

- **Blind Image Denoising and Inpainting Using Robust Hadamard Autoencoders – 2021**

In this paper they demonstrate a novel approach for denoising as well as image inpainting, in which we develop a testable version of the Robust Deep Autoencoders. They have also extended the capabilities of the model to be able to oversee coherent corruptions such as random blocks of missing values in a dataset by imputing them based on the values of the surrounding pixels. This has proven useful for missing value imputation in manufacturing datasets, where we often must deal with random blocks of missing process or chemistry data, and we cannot always use naive imputation techniques such as mean imputation [22]– [24]. Moreover, in such scenarios, we deal with a small-sized dataset and do not want to reduce the size of the data even further by simply eliminating the rows that do not have the complete set of information. Another important application of our model is for imputing and making predictions on graph network data, for example in social networks.

- **A survey on deep learning-based Monte Carlo denoising –** 2021

In general, conventional MC integration approaches perform value estimation through stochastic schemes per footprint, e.g., pixel or shading point. On the other hand, deep learning-based MC denoising can be observed as a complementary postprocessing technique to explore the generality of spatial, temporal, and semantic correlations between rendering footprints and auxiliary features from offline datasets. It is not mandatory, in the conventional sense, but has achieved remarkable success in practice and raised a lot of academic interest by revealing another dimension of the rendering problem, which is influencing in-depth studies and might lead to interesting generation rendering applications in the future. Some of the remaining open frameworks such as Metropolis light transportation, the balance between mathematical convergence and regression efficiency, exploration of novel features and deep-learning models, and improved computation speed for robust real-time rendering. Hopefully, this survey will introduce deep learning-based MC denoising to a large audience and lead to follow-up research in different directions.

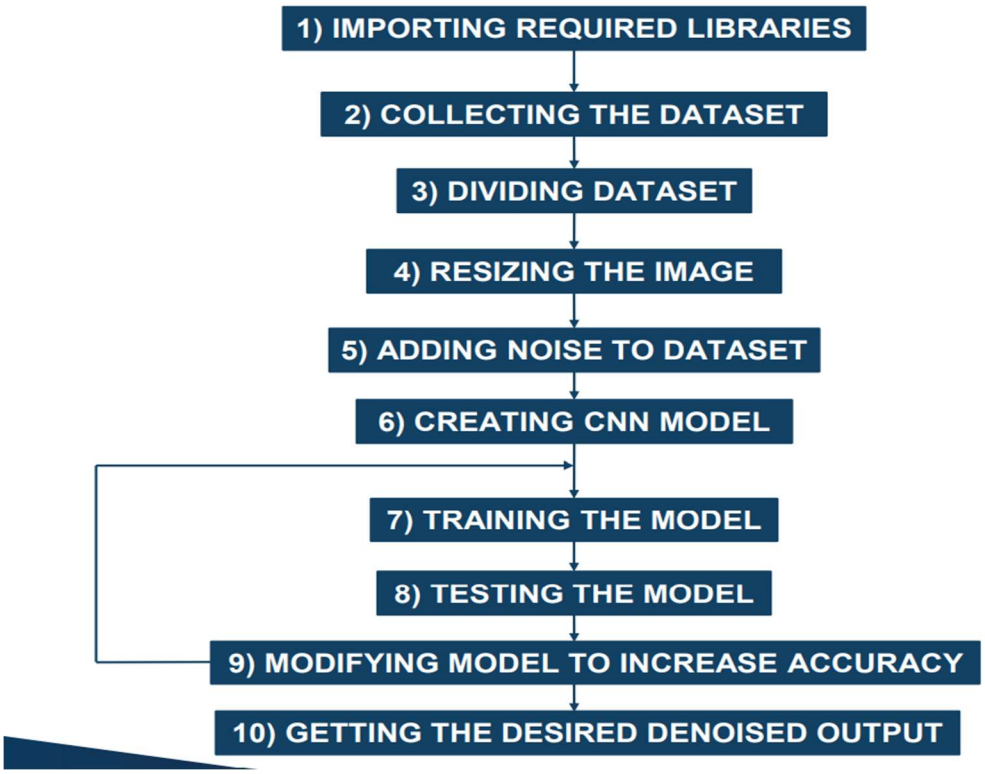
- **An Adaptive Learning Image Denoising Algorithm Based on Eigenvalue Extraction and the GAN Model – 2022**

This paper proposes a self-adjusting generative confrontation network image denoising algorithm. The algorithm includes noise reduction and the adaptive learning GAN model. It pre-processes the image first by combining image characteristics to make the target clear. The threshold value quoted in the process of noise reduction avoids the phenomenon of "over strangling" and improves the effective signal in the high-frequency signal. In the GAN model, iterative updates are performed through operations such as mutation, evaluation, and selection. Self-learning of the generator can quickly and accurately locate the key areas in the image to obtain the optimal value. The results show that the proposed algorithm can effectively denoise while retaining the image signal, which is consistent with the expected effect. However, with the explosive growth of the amount of image data, how to optimize the algorithm to improve the efficiency of image denoising execution is a problem that needs further research.

3. METHODOLOGY

3.1 Steps of Algorithm:

1. Importing Required Libraries
2. Collecting the Dataset
3. Dividing Dataset (Original, Noisy, Training, Testing)
4. Resizing the Image
5. Adding Noise to Dataset
6. Creating CNN model
7. Training the Model
8. Testing the model
9. Modifying Model to Increase Accuracy
10. Getting the Desired Denoised Output



3.2 Proposed Method:

Since CNN using with the help of encoders and decoders have shown satisfactory results. We will be using auto encoders and auto decoders using relu, sigmoid and Adam to find the best fit and to train the efficient model using (error and trial method) using data sets of COCO, Kaggle, MIT-Adobe, etc.

- https://cocodataset.org/#explore
- https://www.kaggle.com/
- https://data.csail.mit.edu/graphics/fivek/
- https://brainweb.bic.mni.mcgill.ca/brainweb/

3.3 Architecture Diagram:

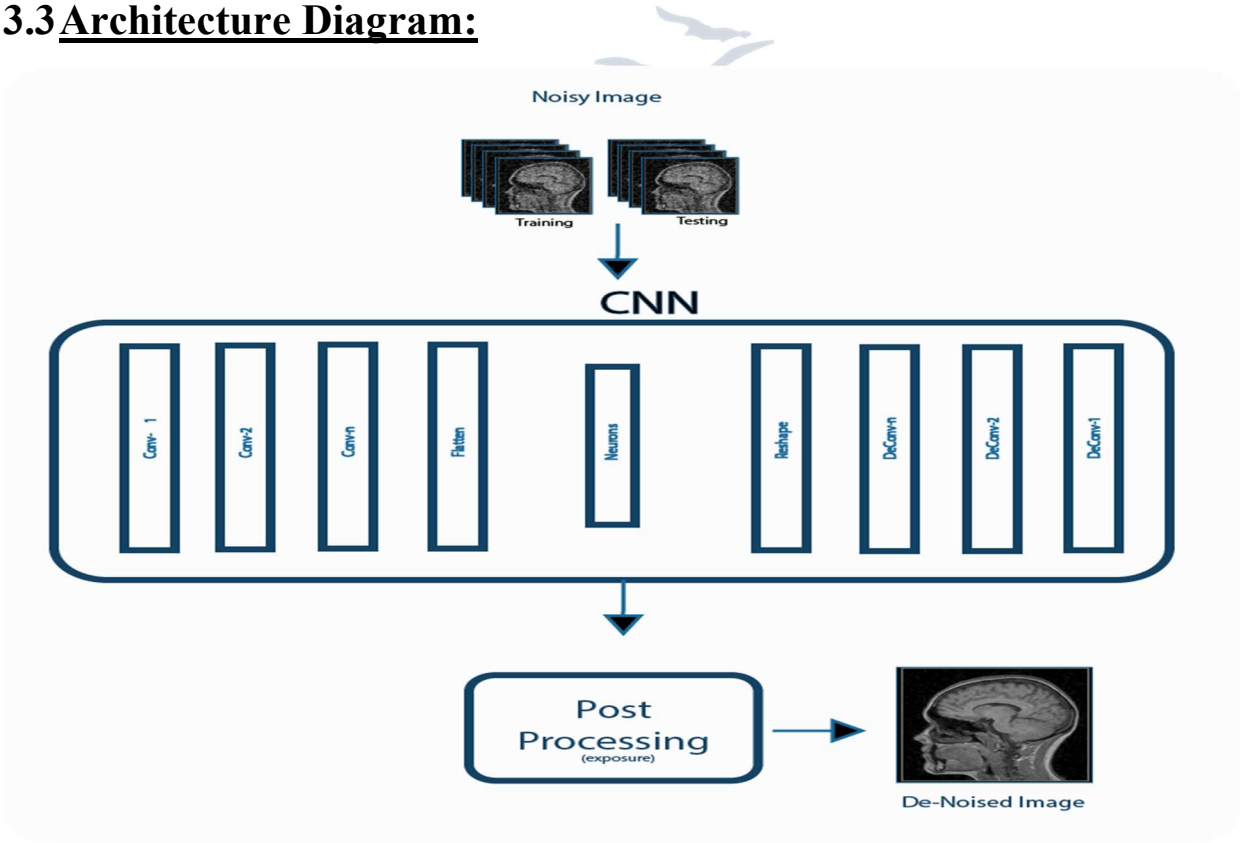


Fig 8. Architecture Diagram

The first step in the Architecture Diagram is to separate the picture dataset into four parts: training the original/denoised image, training the noisy image, evaluating the original/denoised image, and testing the noisy image. Original

training testing dataset and original noisy testing dataset are two datasets for training and two datasets for testing. These are the four datasets we will use, and we will use convolutional layers within the CNN method. It might be anything from one to N, and convolution layers are used. We will use N number of convolution layers and N number of max-pulling layers once we have used convolution layers. After that, we will flatten out all the image's pixels into a 1D array and feed it to the neurons to train the model. After training the neurons, we will take whatever output the model produces and alter it. We flattened it first, and now we will return to the 2D picture format because the image will be in 2D on the X and Y axes. So, we will reshape them before applying deconvolution layers. We have done N convolution layers, and we will do the same number of deconvolution layers. We will undertake pre-processing after the image have been extracted from the model, which will include adding or deleting exposure, as well as other tasks. We will then store it to a hard drive or computer.

3.4 Existing methodology

Gaussian

```
from skimage import img_as_float
from skimage. metrics import peak_signal_noise_ratio
from matplotlib import pyplot as plt
from skimage import io
from scipy import ndimage as nd
import cv2

noisy_img = img_as_float (cv2.imread("Mnoiseimg.tiff",0))
#Need to convert to float as we will be doing math on the array
#Also, most skimage functions need float numbers
ref_img = img_as_float(cv2.imread("Mrealimg.tiff",0))

gaussian_img = nd.gaussian_filter(noisy_img, sigma=5)
plt.imshow(gaussian_img, cmap='gray')
plt.imsave("Gaussian_smoothed.tiff", gaussian_img, cmap='gray')

noise_psnr = peak_signal_noise_ratio(ref_img, noisy_img)
gaussian_cleaned_psnr = peak_signal_noise_ratio(ref_img, gaussian_img)
print("PSNR of input noisy image = ", noise_psnr)
print("PSNR of cleaned image = ", gaussian_cleaned_psnr)
```

Bilateral, TV(total-variation) and Wavelet

```
from skimage.restoration import (denoise_tv_chambolle, denoise_bilateral,
                                 denoise_wavelet, estimate_sigma)
from skimage import img_as_float

#Noisy_img = img_as_float(io. imread("noiseimg.tiff",0))
sigma_est = estimate_sigma(noisy_img, multichannel=False, average_sigmas=True)

denoise_bilateral = denoise_bilateral(noisy_img, sigma_spatial=15,multichannel=False)

noise_psnr = peak_signal_noise_ratio(ref_img, noisy_img)
bilateral_cleaned_psnr = peak_signal_noise_ratio(ref_img, denoise_bilateral)

print("PSNR of input noisy image = ", noise_psnr)
print("PSNR of cleaned image = ", bilateral_cleaned_psnr)
plt.imsave("bilateral_smoothed.tiff", denoise_bilateral, cmap='gray')

denoise_TV = denoise_tv_chambolle(noisy_img, weight=0.3, multichannel=False)
noise_psnr = peak_signal_noise_ratio(ref_img, noisy_img)
TV_cleaned_psnr = peak_signal_noise_ratio(ref_img, denoise_TV)
print("PSNR of input noisy image = ", noise_psnr)
print("PSNR of cleaned image = ", TV_cleaned_psnr)
plt.imshow(denoise_TV, cmap='gray')
plt.imsave("TV_smoothed.tiff", denoise_TV, cmap='gray')


wavelet_smoothed = denoise_wavelet(noisy_img,
multichannel=False,method='BayesShrink', mode='soft', rescale_sigma=True)
noise_psnr = peak_signal_noise_ratio(ref_img, noisy_img)
Wavelet_cleaned_psnr = peak_signal_noise_ratio(ref_img, wavelet_smoothed)
print("PSNR of input noisy image = ", noise_psnr)
print("PSNR of cleaned image = ", Wavelet_cleaned_psnr)
plt.imshow(wavelet_smoothed, cmap='gray')
plt.imsave("wavelet_smoothed.tiff", wavelet_smoothed, cmap='gray',)
```

Shift invariant wavelet denoising

```
import matplotlib.pyplot as plt

from skimage.restoration import denoise_wavelet, cycle_spin
from skimage import data, img_as_float
```

```
from skimage.util import random_noise
from skimage.metrics import peak_signal_noise_ratio
from skimage import io



denoise_kwargs = dict(multichannel=False, wavelet='db1', method='BayesShrink',
                      rescale_sigma=True)

all_psnr = []
max_shifts = 3     #0, 1, 3, 5

Shft_inv_wavelet = cycle_spin(noisy_img, func=denoise_wavelet, max_shifts = max_shifts,
                          func_kw=denoise_kwargs, multichannel=False)

noise_psnr = peak_signal_noise_ratio(ref_img, noisy_img)
shft_cleaned_psnr = peak_signal_noise_ratio(ref_img, Shft_inv_wavelet)

print("PSNR of input noisy image = ", noise_psnr)
print("PSNR of cleaned image = ", shft_cleaned_psnr)

plt.imsave("Shift_Inv_wavelet_smoothed.tiff", Shft_inv_wavelet, cmap='gray')


print(wavelet_smoothed.dtype)
```

Anisotropic Diffusion


```
import matplotlib.pyplot as plt
import cv2
from skimage import io
from medpy.filter.smoothing import anisotropic_diffusion
from skimage import img_as_float
from skimage.metrics import peak_signal_noise_ratio


#img = io.imread("MRI_images/MRI_noisy.tif", as_gray=True)
#Noisy_img = img_as_float (io.imread("noiseimg.tiff", as_gray=True))
#Ref_img = img_as_float(io. imread("realimg.tiff"))

# niter= number of iterations
#kappa = Conduction coefficient (20 to 100)
#gamma = speed of diffusion (<=0.25)
#Option: Perona Malik equation 1 or 2. A value of 3 is for Turkey's biweight function
img_aniso_filtered = anisotropic_diffusion(noisy_img, niter=50, kappa=50, gamma=0.2,
option=2)
```

```
noise_psnr = peak_signal_noise_ratio(ref_img, noisy_img)
anisotropic_cleaned_psnr = peak_signal_noise_ratio(ref_img, img_aniso_filtered)
print("PSNR of input noisy image = ", noise_psnr)
print("PSNR of cleaned image = ", anisotropic_cleaned_psnr)


plt.imshow(img_aniso_filtered, cmap='gray')
plt.imsave("anisotropic_denoised.tiff", img_aniso_filtered, cmap='gray')

from skimage.restoration import denoise_nl_means, estimate_sigma
from skimage import img_as_ubyte, img_as_float
from matplotlib import pyplot as plt
from skimage import io
import numpy as np
from skimage.metrics import peak_signal_noise_ratio
#Noisy_img = img_as_float (io.imread("images/MRI_images/MRI_noisy.tif",
as_gray=True))
#Ref_img = img_as_float(io. imread("images/MRI_images/MRI_clean.tif"))

sigma_est = np.mean(estimate_sigma(noisy_img, multichannel=False))


NLM_skimg_denoise_img = denoise_nl_means(noisy_img, h=1.15 * sigma_est,
fast_mode=True,
                    patch_size=9, patch_distance=5, multichannel=False)

noise_psnr = peak_signal_noise_ratio(ref_img, noisy_img)
NLM_skimg_cleaned_psnr = peak_signal_noise_ratio(ref_img, NLM_skimg_denoise_img)
print("PSNR of input noisy image = ", noise_psnr)
print("PSNR of cleaned image = ", NLM_skimg_cleaned_psnr)


denoise_img_as_8byte = img_as_ubyte(NLM_skimg_denoise_img)

#Plt.imshow(NLM_skimg_denoise_img)
#Plt.imshow(denoise_img_as_8byte, cmap=plt.cm. gray, interpolation='nearest')
plt.imsave("NLM_skimage_denoised.tiff", denoise_img_as_8byte, cmap='gray')
```

BM3D

```
import matplotlib.pyplot as plt
from skimage import io, img_as_float
from skimage.metrics import peak_signal_noise_ratio
import bm3d
```

```
import numpy as np

#Noisy_img = img_as_float (io.imread("images/MRI_images/MRI_noisy.tif",
as_gray=True))
#Ref_img = img_as_float(io. imread("images/MRI_images/MRI_clean.tif"))


BM3D_denoised_image = bm3d.bm3d(noisy_img, sigma_psd=0.2,
stage_arg=bm3d.BM3Dstages.ALL_STAGES)
#BM3D_denoised_image = bm3d.bm3d(noisy_img, sigma_psd=0.2,
stage_arg=bm3d.BM3Dstages.HARD_THRESHOLDING)

 #Also try stage_arg=bm3d.BM3Dstages.HARD_THRESHOLDING


noise_psnr = peak_signal_noise_ratio(ref_img, noisy_img)
BM3D_cleaned_psnr = peak_signal_noise_ratio(ref_img, BM3D_denoised_image)
print("PSNR of input noisy image = ", noise_psnr)
print("PSNR of cleaned image = ", BM3D_cleaned_psnr)


plt.imshow(BM3D_denoised_image, cmap='gray')
plt.imsave("BM3D_denoised.tiff", BM3D_denoised_image, cmap='gray')
```

3.5 Dataset Images and Proposed Code of Denoising Image

Modules:

```
In [5]: import numpy as np
        import tensorflow as tf
        import matplotlib.pyplot as plt
        import os
        from tensorflow.keras import layers
        from tensorflow.keras.datasets import mnist
        from tensorflow.keras.models import Model
        from IPython.display import Image
        from PIL import Image
        import brainweb
        from brainweb import volshow
        import numpy as np
        from os import path
        from tqdm.auto import tqdm
        import logging
        logging.basicConfig(level=logging.INFO)
        import matplotlib.pyplot as plt
        import cv2
        import imageio
        import math
        from skimage import color
        from skimage import io
        import random
        random.seed(0)
        from skimage.transform import resize
```

Importing Images

```
files = brainweb.get_files()


data = brainweb.load_file(files[0])
```

Training Denoise/Original Images

```
In [7]: #3
        for i in range(1,10):

            data1 = brainweb.load_file(files[i])
            data8=[]
            for images in data1:
                res = resize(images, (400, 400))
                data8.append(res)
            data1=np.array(data8)
            data1 = data1.astype("float32") / 255.0
            data1 = np.reshape(data1,(len(data1), 400, 400, 1))
            clean_data= np.concatenate((clean_data,data1))
```

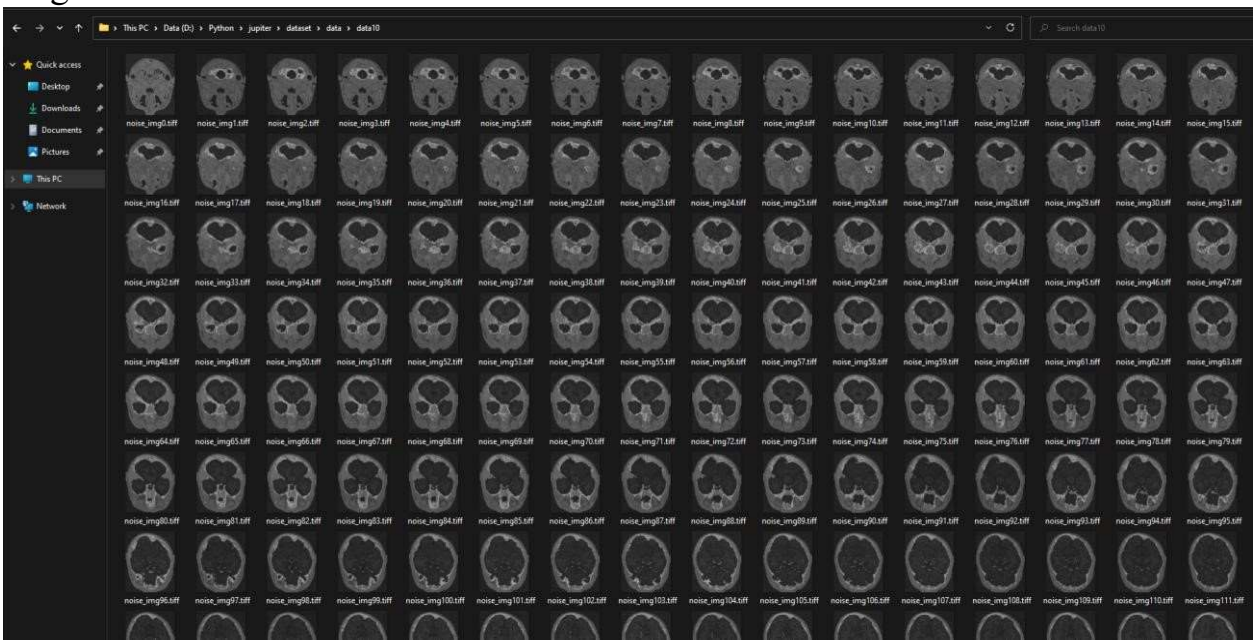
Adding Noise to Images

```
In [9]: array2=[]
        for img in data1:
            noise=random.randint(0, 45)
            print("noise="+str(noise))
            ni=np.add(np.random.normal(0, noise,size=img.shape),img)
            nj=np.add(np.random.normal(0, noise,size=img.shape),img)
            noi=np.sqrt(np.multiply(ni,ni)+np.multiply(nj,nj))
            noi=resize(noi, (400, 400))
            noi = noi.astype("float32") / 255.0
            array2.append(noi)

        noi_array=np.array(array2)
```

Saving the Processed Image

```
In [12]: x=0
         for images in test_noi_data:
             print(images.shape)
             plt.imshow(images,cmap='gray')
             plt.imsave("dataset/data/testnoise/img"+str(x)+".tiff", images,cmap='gray')
             x=x+1
             plt.show()
```


Images datasets



Loading Datasets

In [2]:
```
import glob
import cv2

train_data = np.array([cv2.imread(file,0) for file in glob.glob("dataset/data/cleanimg/*.tiff")])
```

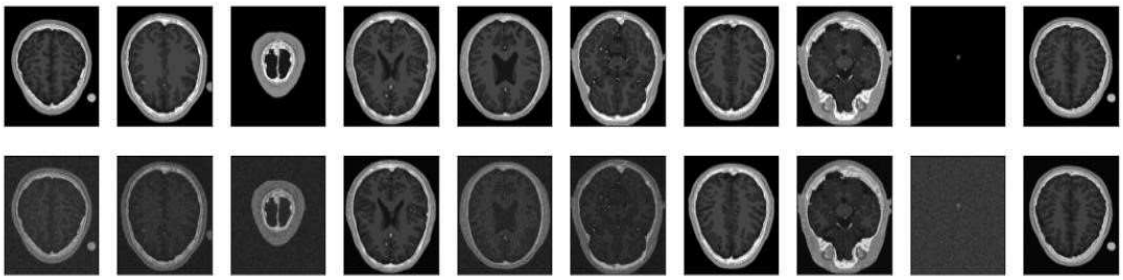
In [3]:
```
test_data= np.array([cv2.imread(file,0) for file in glob.glob("dataset/data/testclean/*.tiff")])
```

In [4]:
```
noisy_train_data= np.array([cv2.imread(file,0) for file in glob.glob("dataset/data/noisytrain/*.tiff")])
```

In [5]:
```
noisy_test_data= np.array([cv2.imread(file,0) for file in glob.glob("dataset/data/testnoise/*.tiff")])
```

Processing Data

In [8]:
```
# Normalize and reshape the data
train_data = preprocess(train_data)
test_data = preprocess(test_data)

# Create a copy of the data with added noise
noisy_train_data = preprocess(noisy_train_data)
noisy_test_data = preprocess(noisy_test_data)

# Display the train data and a version of it with added noise
display(train_data, noisy_train_data)
```



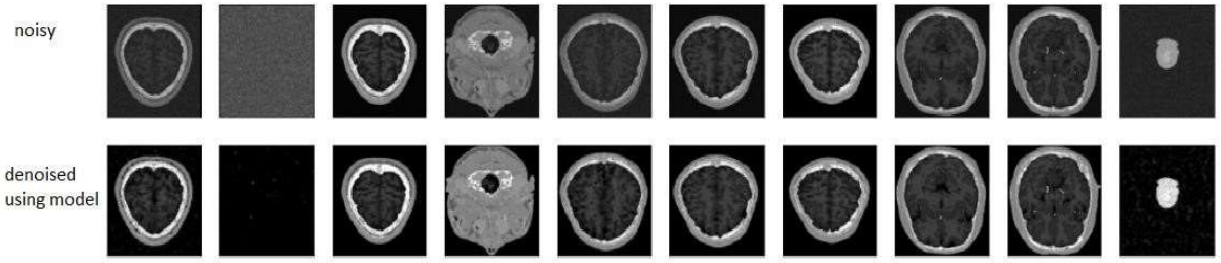
Proposed Model

```
input = layers.Input(shape=(400, 400, 1))

# Encoder
x = layers.Conv2D(32, (3, 3), activation="relu", padding="same")(input)
x = layers.MaxPooling2D((2, 2), padding="same")(x)
x = layers.Conv2D(32, (3, 3), activation="relu", padding="same")(x)
x = layers.MaxPooling2D((2, 2), padding="same")(x)

# Decoder
x = layers.Conv2DTranspose(32, (3, 3), strides=2, activation="relu", padding="same")(x)
x = layers.Conv2DTranspose(32, (3, 3), strides=2, activation="relu", padding="same")(x)
x = layers.Conv2D(1, (3, 3), activation="sigmoid", padding="same")(x)

# Autoencoder
autoencoder = Model(input, x)
autoencoder.compile(optimizer="adam", loss="binary_crossentropy")
autoencoder.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 400, 400, 1)]     0

 conv2d (Conv2D)             (None, 400, 400, 32)      320

 max_pooling2d (MaxPooling2D  (None, 200, 200, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 200, 200, 32)      9248

 max_pooling2d_1 (MaxPooling  (None, 100, 100, 32)     0
 2D)

 conv2d_transpose (Conv2DTra  (None, 200, 200, 32)     9248
 nspose)

 conv2d_transpose_1 (Conv2DT  (None, 400, 400, 32)     9248
 ranspose)

 conv2d_2 (Conv2D)           (None, 400, 400, 1)       289

=================================================================
Total params: 28,353
Trainable params: 28,353
Non-trainable params: 0
_________________________________________________________________
```

Output of Proposed Model

```
In [15]: predictions = autoencoder.predict(noisy_test_data)
         display(noisy_test_data, predictions)
```

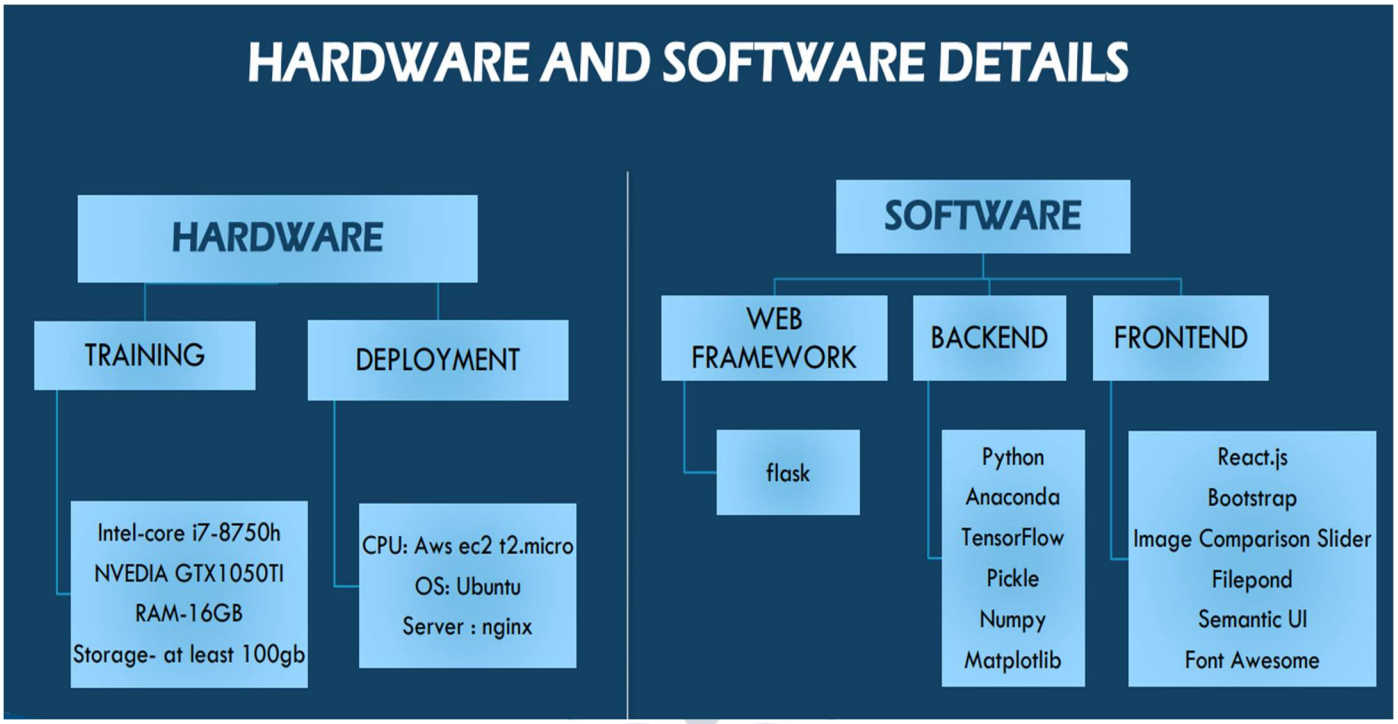


Model PSNR at beginning

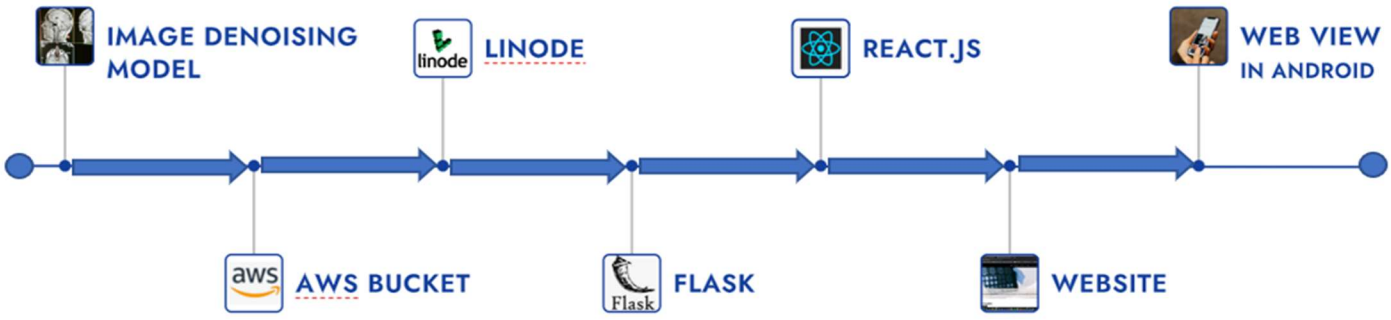
```
In [3]: from skimage import img_as_float
        from skimage.metrics import peak_signal_noise_ratio
        from matplotlib import pyplot as plt
        from skimage import io
        from scipy import ndimage as nd
        import cv2

In [7]: noisy_img = img_as_float(cv2.imread("Mnoiseimg.tiff",0))
        #Need to convert to float as we will be doing math on the array
        #Also, most skimage functions need float numbers
        ref_img = img_as_float(cv2.imread("Mrealimg.tiff",0))
        predicted=img_as_float(cv2.imread("outfile3.tiff",0))

In [8]: noise_psnr = peak_signal_noise_ratio(ref_img, noisy_img)
        gaussian_cleaned_psnr = peak_signal_noise_ratio(ref_img, predicted)
        print("PSNR of input noisy image = ", noise_psnr)
        print("PSNR of cleaned image = ", gaussian_cleaned_psnr)

        PSNR of input noisy image =  22.219345330569695
        PSNR of cleaned image =  27.14978261896217

In [ ]:
```

4. REQUIREMENTS ANALYSIS

4.1 Hardware and Software Requirements



4.2 Flow of Process



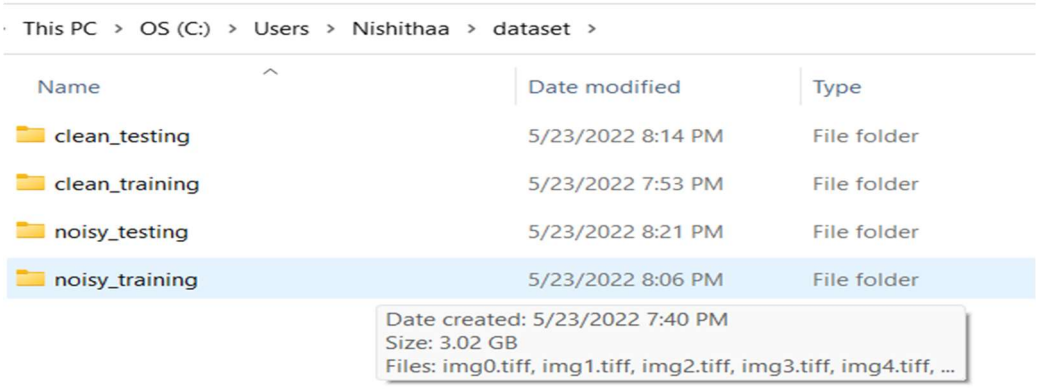
5. IMPLEMENTATION

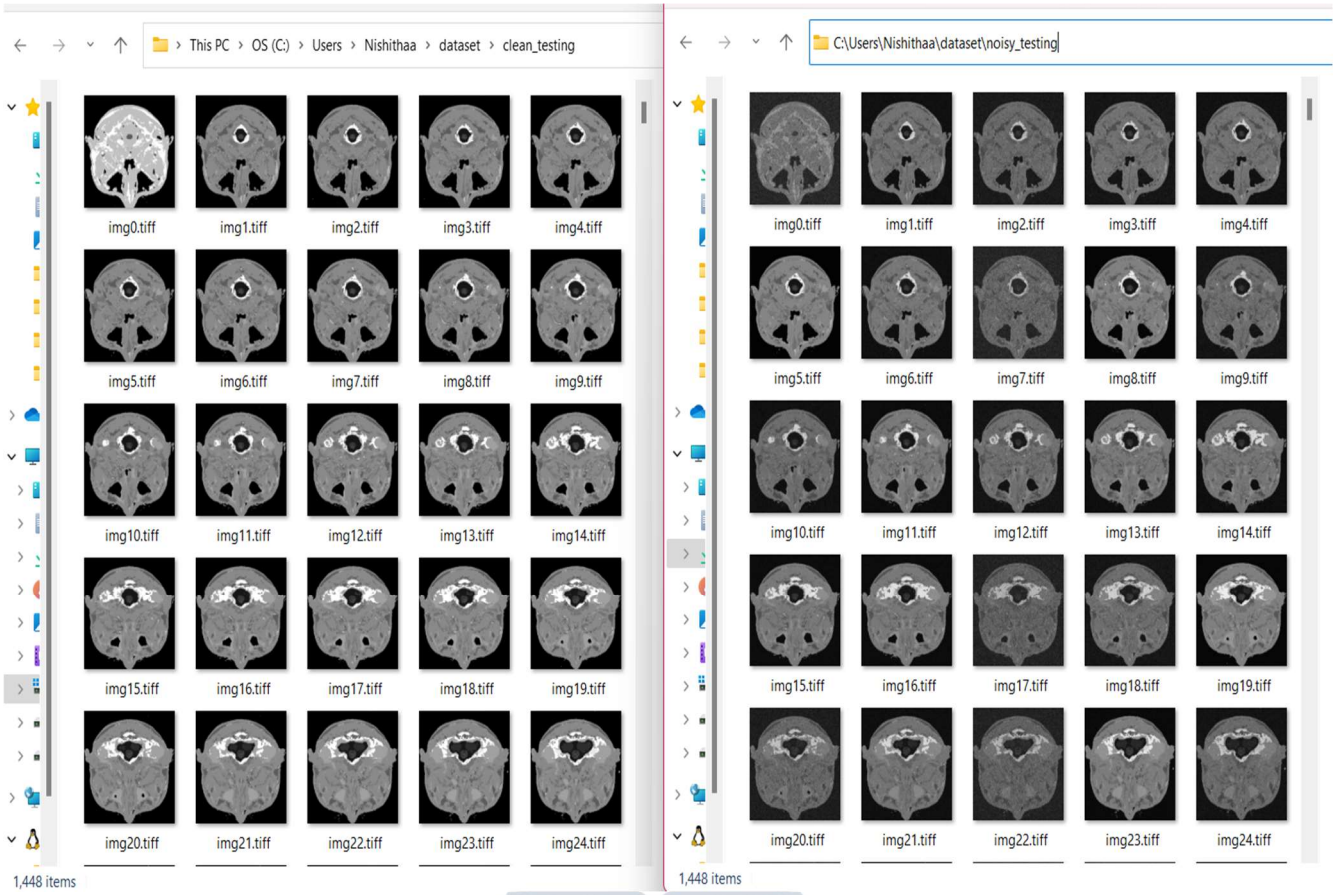
5.1 Dataset creation

```
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import os
from tensorflow.keras import layers
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Model
from IPython.display import Image
from PIL import Image
import brainweb
from brainweb import volshow
import numpy as np
from os import path
from tqdm.auto import tqdm
import logging
logging.basicConfig(level=logging.INFO)
import matplotlib.pyplot as plt
import cv2
import imageio
import math
from skimage import color
from skimage import io
import random
random.seed(0)
from skimage.transform import resize
def display(array1, array2):
    """
    Displays ten random images from each one of the supplied arrays.
    """
  n = 10

    indices = np.random.randint(len(array1), size=n)
    images1 = array1[indices, :]
    images2 = array2[indices, :]
```

```
    plt.figure(figsize=(20, 4))
    for i, (image1, image2) in enumerate(zip(images1, images2)):
        ax = plt.subplot(2, n, i + 1)
        plt.imshow(image1.reshape(400, 400))
        plt.gray()
        ax.get_xaxis().set_visible(False)
        ax.get_yaxis().set_visible(False)

        ax = plt.subplot(2, n, i + 1 + n)
        plt.imshow(image2.reshape(400, 400))
        plt.gray()
        ax.get_xaxis().set_visible(False)
        ax.get_yaxis().set_visible(False)

    plt.show()

files = brainweb.get_files()
for i in range(1,10):

    data1 = brainweb.load_file(files[i])
    data8=[]
    for images in data1:
        res = resize(images, (400, 400))
        data8.append(res)
    data1=np.array(data8)
    data1 = data1.astype("float32") / 255.0
    data1 = np.reshape(data1,(len(data1), 400, 400, 1))
    clean_data= np.concatenate((clean_data,data1))

x=0
for images in clean_data:
    print(images.shape)
    plt.imshow(images,cmap='gray')
    plt.imsave("dataset/data/testnoise/img"+str(x)+".tiff", images,cmap='gray')
    x=x+1
    plt.show()


files = brainweb.get_files()
```

```
# read last file
data1 = brainweb.load_file(files[0])

for i in range(1,10):
    data12 = brainweb.load_file(files[i])
    data1= np.concatenate((data1,data12))

array2=[]
for img in data1:
    noise=random.randint(0, 45)
    print("noise="+str(noise))
    ni=np.add(np.random.normal(0, noise,size=img.shape),img)
    nj=np.add (np.random.normal(0, noise,size=img. shape), img)
    noi=np. sqrt(np.multiply(ni,ni)+np.multiply(nj,nj))
    noi=resize (noi, (400, 400))
    noi = noi. astype("float32") / 255.0
    array2.append(noi)

noi_array=np. array(array2)

noi_array1 = np. reshape(noi_array,(len(noi_array), 400, 400))

display (clean_data, noi_array1)
```



5.2 Image Denoising Model

Now let us see how the model is trained and its accuacy when compared to other methods and model in order to find their accuracy we have used PSNR values.

- Firstly, the model the model contans of ENCODER, DECODER and AUTOENCODER.
- In ENCODER we used convolution2D layers with activation relu and pading as same we have used 2 layes of this layer along with MAX pooling 2D with pading.
- 128 hidden layers with activation relu is included in ENCODER.

- In DECODING we used convolution2Dtranspose layers with activation relu and pading as same.
- In AUTOENCODER to compile we have used optimizer adam and loss function as binary_crossentropy and the model has given promising results.

❖ original PSRN value of noisy image: **22.21**

❖ the accuracy with our model and other model are as follows:

FILTER/METHOD	PSNR
gaussian_filter	18.08
denoise_bilateral	18.22
denoise_wavelet	22.43
Shft_inv_wavelet	22.46
img_aniso_filter	18.36
denoise_img_as_8byte	22.63
BM3D_denoise	21.88
previous_model	27.15
present_model	**28.79**

5.3 Presenting Our Model

```
SIZE = 400
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=(SIZE, SIZE, 1)))
model.add(MaxPooling2D((2, 2), padding='same'))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D((2, 2), padding='same'))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D((2, 2), padding='same'))
model.add(Dense(128, activation='relu'))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(UpSampling2D((2, 2)))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(UpSampling2D((2, 2)))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(UpSampling2D((2, 2)))
model.add(Conv2D(1, (3, 3), activation='relu', padding='same'))

model.compile(optimizer='adam', loss='mean_squared_error', metrics=['accuracy'])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 400, 400, 32)      320

 max_pooling2d (MaxPooling2D  (None, 200, 200, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 200, 200, 32)      9248

 max_pooling2d_1 (MaxPooling  (None, 100, 100, 32)     0
 2D)

 conv2d_2 (Conv2D)           (None, 100, 100, 32)      9248

 max_pooling2d_2 (MaxPooling  (None, 50, 50, 32)       0
 2D)

 dense (Dense)               (None, 50, 50, 128)       4224

 conv2d_3 (Conv2D)           (None, 50, 50, 32)        36896

 up_sampling2d (UpSampling2D  (None, 100, 100, 32)     0
 )

 conv2d_4 (Conv2D)           (None, 100, 100, 32)      9248

 up_sampling2d_1 (UpSampling  (None, 200, 200, 32)     0
 2D)

 conv2d_5 (Conv2D)           (None, 200, 200, 32)      9248

 up_sampling2d_2 (UpSampling  (None, 400, 400, 32)     0
 2D)

 conv2d_6 (Conv2D)           (None, 400, 400, 1)       289

=================================================================
Total params: 78,721
Trainable params: 78,721
Non-trainable params: 0
_________________________________________________________________
```

5.4 Frontend Code [ReactJs]

Index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8" />
  <link rel="icon" href="images/logo2.png" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <meta name="theme-color" content="#000000" />
  <meta name="description"
    content="CNN-based Image Denoising Model implemented with the help of
encoders and decoders to estimate the original image by suppressing noise
from a noise contaminated version of the image by applying our efficient
algorithm to denoise the image with the highest SSIM and PSNR values with
low processing time. The concept of maxpooling2D and convo2D layer, is
implied to create autoencoders and decoders. Where in the model, Relu and
Sigmoid are implemented as activation functions and Adam is used as an
optimizer for autoencoders." />
  <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min
.css" rel="stylesheet"
    integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">
  <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
  <link href="https://use.fontawesome.com/releases/v5.15.1/css/all.css"
rel="stylesheet" />
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css">

  <!-- CSS FOR ACCORDION -->
  <link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Roboto:400,500|Open+Sans">
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.mi
n.css">
  <link rel="stylesheet"
href="https://fonts.googleapis.com/icon?family=Material+Icons">
```

```
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css">
  <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
  <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js
"></script>
  <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.
js"></script>



  <!-- Image Comparison -->
  <script defer src="https://unpkg.com/img-comparison-
slider@7/dist/index.js"></script>
  <link rel="stylesheet" href="https://unpkg.com/img-comparison-
slider@7/dist/styles.css" />
  <title>Medical Image Denoising</title>

  <!-- Accordion -->
  <link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Roboto:400,500|Open+Sans">
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.mi
n.css">
  <link rel="stylesheet"
href="https://fonts.googleapis.com/icon?family=Material+Icons">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css">

</head>

<body>
  <noscript>You need to enable JavaScript to run this app.</noscript>
  <div id="root"></div>

  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundl
e.min.js"
    integrity="sha384-
MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
    crossorigin="anonymous"></script>
  <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
    type="607d0eab00111e6864de75e1-text/javascript"></script>
```

```
  <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js
"
    type="607d0eab00111e6864de75e1-text/javascript"></script>
  <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.
js"
    type="607d0eab00111e6864de75e1-text/javascript"></script>
  <script src="/cdn-cgi/scripts/7d0fa10a/cloudflare-static/rocket-
loader.min.js"
    data-cf-settings="607d0eab00111e6864de75e1-|49" defer=""></script>

  <!-- Accordion -->
  <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
  <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js
"></script>
  <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.
js"></script>

  <!-- Tools -->
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></
script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/slick-
carousel/1.9.0/slick.min.js"></script>
  <script type="text/javascript" src="//code.jquery.com/jquery-
1.11.0.min.js"></script>
  <script type="text/javascript" src="//code.jquery.com/jquery-migrate-
1.2.1.min.js"></script>
  <script type="text/javascript" src="slick/slick.min.js"></script>

  <!-- Footer Section -->
  <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min
.css"></script>
  <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.bund
le.min.js"></script>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"><
/script>
  <script
src="https://use.fontawesome.com/releases/v5.7.2/css/all.css"></script>
```

```
</body>

</body>

</html>
```

Index.js

```
import React from "react";
import reportWebVitals from "./reportWebVitals";
import "./index.css";
import App from "./App";

import "bootstrap/dist/css/bootstrap.min.css";
import "slick-carousel/slick/slick.css";
import "slick-carousel/slick/slick-theme.css";

import "./Components/Team.css";
import "./Components/ImageComparison.css";
import "./Components/Steps.css";
import "./Components/AccordionQuestions.css";
import "./Components/Tools.css";
import "./Components/Timeline.css";
import "./Components/Footer.css";
import "./Components/CarouselHome.css";

import { createRoot } from "react-dom/client";

const rootElement = document.getElementById("root");
const root = createRoot(rootElement);

root.render(<App />);

reportWebVitals();
```

App.js

```
import React from "react";
import AccordionQuestions from "./Components/AccordionQuestions";
import AlgorithmSteps from "./Components/AlgorithmSteps";
import CarouselHome from "./Components/CarouselHome";
import Footer from "./Components/Footer";
```

```
import ImageComparison from "./Components/ImageComparison";
import Navbar from "./Components/Navbar";
import Team from "./Components/Team";
import Timeline from "./Components/Timeline";
import ToolsUsed from "./Components/ToolsUsed";

const App = () => {
  return (
    <>
      <Navbar />
      <CarouselHome />
      <Timeline />
      <AlgorithmSteps />
      <ImageComparison />
      <ToolsUsed />
      <Team />
      <AccordionQuestions />
      <Footer />
    </>
  );
};

export default App;
```


Package.json

```
{
  "name": "image_processing",
  "homepage": "https://mohammedsaif001.github.io/Medical-Image-
Denoising/",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.16.3",
    "@testing-library/react": "^12.1.4",
    "@testing-library/user-event": "^13.5.0",
    "bootstrap": "^5.1.3",
    "filepond": "^4.30.3",
    "filepond-plugin-image-exif-orientation": "^1.0.11",
    "filepond-plugin-image-preview": "^4.6.11",
    "font-awesome": "^4.7.0",
    "gh-pages": "^3.2.3",
    "imagemagick": "^0.1.3",
    "jquery": "^3.6.0",
```

```
    "mdb-react-ui-kit": "^3.0.0",
    "react": "^18.0.0",
    "react-bootstrap": "^2.2.2",
    "react-compare-slider": "^2.2.0",
    "react-dom": "^18.0.0",
    "react-filepond": "^7.1.1",
    "react-image-comparison-slider": "^1.8.4",
    "react-scripts": "^5.0.0",
    "react-slick": "^0.29.0",
    "reactjs-popup": "^2.0.5",
    "semantic-ui-css": "^2.4.1",
    "semantic-ui-react": "^2.1.2",
    "slick-carousel": "^1.8.1",
    "tiff-to-png": "^2.0.2",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "predeploy": "npm run build",
    "deploy": "gh-pages -d build",
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

COMPONENTS OF EACH SECTION OF WEBSITE

Navbar Section:

1. Navbar.js

```
import React, { useEffect } from "react";

import ModalNav from "./ModalNav";

function Navbar() {
  const [modalShow, setModalShow] = React.useState(false);

  useEffect(() => {
    // Collapsible Navbar after Clicking Link
    $(".js-scroll-trigger").on("click", function () {
      $(".navbar-collapse").collapse("hide");
    });
  });

  return (
    <nav
      className="navbar navbar-expand-lg navbar-dark  fixed-top mb-3"
      style={{ background: "black" }}
      id="navigation"
    >
      <div className="container-fluid">
        <a className="navbar-brand" href="#">
          <img
            src="images/logo2.png"
            width="30"
            height="30"
            className="d-inline-block align-top"
            alt=""
          />{" "}
          Medical Image Denoise
        </a>
        <button
          className="navbar-toggler"
          type="button"
          data-bs-toggle="collapse"
          data-bs-target="#navbarSupportedContent"
          aria-controls="navbarSupportedContent"
          aria-expanded="false"
          aria-label="Toggle navigation"
        >
          <span className="navbar-toggler-icon"></span>
        </button>
```

```
        <div className="collapse navbar-collapse"
id="navbarSupportedContent">
          <ul className="navbar-nav ms-auto mb-2 mb-lg-0">
            <li className="nav-item ">
              <a
                className="nav-link  js-scroll-trigger "
                aria-current="page"
                href="#"
              >
                Home
              </a>
            </li>
            <li className="nav-item ">
              <a className="nav-link js-scroll-trigger" href="#research">
                Research
              </a>
            </li>
            <li className="nav-item ">
              <a className="nav-link js-scroll-trigger" href="#algo">
                Algorithm
              </a>
            </li>
            <li className="nav-item">
              <a className="nav-link js-scroll-trigger" href="#working">
                Working
              </a>
            </li>
            <li className="nav-item">
              <a className="nav-link js-scroll-trigger" href="#tools">
                Tools Used
              </a>
            </li>
            <li className="nav-item">
              <a className="nav-link js-scroll-trigger" href="#aboutus">
                About Us
              </a>
            </li>
            <li className="nav-item">
              <a className="nav-link js-scroll-trigger" href="#faq">
                FAQs
              </a>
            </li>

            <button
              className="btn btn-outline-light my-2 my-sm-0 mx-3"
```

```
              type="submit"
              onClick={() => setModalShow(true)}
            >
              Upload
            </button>
          </ul>

          <ModalNav show={modalShow} onHide={() => setModalShow(false)}
/>
        </div>
      </div>
    </nav>
  );
}

export default Navbar;
```

Carousel Section:

1. **CarouselHome.js**

```
import React from 'react'
import Carousel from 'react-bootstrap/Carousel'

function CarouselHome() {
    return (
        <div >
            <Carousel variant="dark">
                <Carousel.Item>
                    <img
                        className="d-block w-100"
                        src="https://images.pexels.com/photos/4226264/pex
els-photo-4226264.jpeg?auto=compress&cs=tinysrgb&w=1260&h=750&dpr=1"
                        alt="First slide"
                    />
                    <Carousel.Caption className='thirdSlide'>
                        <h5>Medical Image Denoising</h5>
                        <p>We denoise medical MRI images since it is a
mandatory and essential preprocessing technique for further medical image
processing stages.</p>
                    </Carousel.Caption>
                </Carousel.Item>
                <Carousel.Item>
                    <img
```

```
                        className="d-block w-100"
                        src="https://images.pexels.com/photos/4226219/pex
els-photo-4226219.jpeg?auto=compress&cs=tinysrgb&w=1260&h=750&dpr=1"
                        alt="Second slide"
                    />
                    <Carousel.Caption className='thirdSlide'>
                        <h5 >What is Image Denoising? </h5>
                        <p > One of the fundamental challenges in the
field of image processing and computer vision is image denoising, where
the underlying goal is to estimate the original image by suppressing
noise from a noise-contaminated version of the image.
                        </p>
                    </Carousel.Caption>
                </Carousel.Item>
                <Carousel.Item>
                    <img
                        className="d-block w-100"
                        src="https://images.pexels.com/photos/5723883/pex
els-photo-5723883.jpeg?auto=compress&cs=tinysrgb&w=1260&h=750&dpr=1"
                        alt="Third slide"
                    />
                    <Carousel.Caption className='thirdSlide'>
                        <h5>Why CNN for Image Denoising? </h5>
                        <p>The CNN based image denoising models have
shown improvement in denoising performance as compared to non-CNN
methods.The use of CNN is not limited to general image denoising alone,
CNN produced excellent results for blind denoising , real noisy images
and many others.</p>
                    </Carousel.Caption>
                </Carousel.Item>
            </Carousel>

        </div>
    )
}

export default CarouselHome
```

2. CarouselHome.css

```
.thirdSlide {
  background-color: rgba(234, 227, 227, 0.807);
  border-radius: 25px;
  padding: 20px 39px 10px 39px;
  /* font-size: 0.5rem; */
```

```
}

@media (max-width: 600px) {
  .thirdSlide {
    background-color: rgba(234, 227, 227, 0.807);
    border-radius: 55px;
    font-size: 75%;
    border-radius: 18px;
    padding: 20px 9px 10px 9px;
  }
  .thirdSlide h5 {
    font-size: 16px;
  }
  .thirdSlide p {
    font-size: 85%;
  }
}
@media (max-width: 450px) {
  .thirdSlide {
    background-color: rgba(234, 227, 227, 0.807);
    border-radius: 55px;
    font-size: 75%;
    border-radius: 15px;
    padding: 20px 9px 10px 9px;
  }
  .thirdSlide h5 {
    font-size: 15px;
  }
  .thirdSlide p {
    font-size: 85%;
  }
}
```

Research Section:

1. **TimelineContent.js**

```
const researchInfo = [
    {
        slNo: "1",
        cssClass: "container4 left",
```

```
        date: "SEPT 2019",
        title:
            "Optimal Bilateral Filter and Convolutional Neural Network
Based Denoising Method of Medical Image Measurements (MEASUR 6587)",
        point1:
            "Bioinspired optimization based filtering technique for the
MI Denoising.",
        point2:
            "Swarm-based optimization (DF and MFF algorithm).",
        point3:
            "CNN is used to classify the denoised image as normal or
abnormal.",
        link:
"https://www.sciencedirect.com/science/article/abs/pii/S0263224119303902?
via%3Dihub",
    },
    {
        slNo: "2",
        cssClass: "container4 right",
        date: "JUL 2020",
        title:
            "A Convolutional Neural Network for Denoising of Magnetic
Resonance Images (PATREC 7847)",
        point1: "CNN-DMRI Model for reduction of Rician noise from MRI
Images.",
        point2:
            "Multiple convolutions captures different image features
while separating inherent noise.",
        point3:
            "Performs Down-sampling and Up-sampling through the Encoder-
Decoder framework.",
        link:
"https://www.sciencedirect.com/science/article/abs/pii/S0167865520301203?
via%3Dihub",
    },
    {
        slNo: "3",
        cssClass: "container4 left",
        date: "SEPT 2020",
        title:
            "MRI Denoising Using Progressively Distribution-Based Neural
Network (MRI 9412)",
        point1:
            "Progressive learning strategy applied to MR Image Rician
Denoising.",
```

```
        point2:
            "Fitting the distribution at pixel-level and feature-level
were performed.",
        point3:
            "Nets was proposed for achieving a better performance.",
        link:
"https://www.sciencedirect.com/science/article/abs/pii/S0730725X19304643?
via%3Dihub",
    },
    {
        slNo: "4",
        cssClass: "container4 right",
        date: "JAN 2021",
        title:
            "Blind Image Denoising and Inpainting Using Robust Hadamard
Autoencoders",
        point1: "Denoising and Image Inpainting - Robust Deep
Autoencoders.",
        point2:
            "Oversee coherent corruptions - Random blocks of missing
values in a dataset.",
        point3:
            "Imputing and making predictions on graph network data.",
        link:
"https://www.researchgate.net/publication/348802795_Blind_Image_Denoising
_and_Inpainting_Using_Robust_Hadamard_Autoencoders",
    },
    {
        slNo: "5",
        cssClass: "container4 left",
        date: "JUN 2021",
        title:
            "Methods for Image Denoising Using Convolutional Neural
Network: A Review",
        point1: "Survey of different techniques relating to CNN Image
Denoising.",
        point2:
            "Focuses on CNN architectures.",
        point3:
            "Observed that the GAN was the most used method for CNN Image
Denoising",
        link: "https://link.springer.com/article/10.1007/s40747-021-
00428-4",
    },
    {
```

```
        slNo: "6",
        cssClass: "container4 right",
        date: "OCT 2021",
        title:
            "Wavelet Enabled Convolutional Autoencoder Based Deep Neural
Network for Hyperspectral Image Denoising",
        point1: "Dual branch DL based denoising method - WaCAEN.",
        point2:
            "CNN, Autoencoder, skip connections and sub-pixel up sampling
for better outcomes.",
        point3:
            "Demonstrates its effectiveness in restoration of spectral
signature.",
        link: "https://link.springer.com/article/10.1007/s11042-021-
11689-z",
    },
];

export default researchInfo;
```

2. Timeline.js

```
import React from "react";
import researchInfo from "./TimelineContent.js";
import TimelineResearch from "./TimelineResearch.js";

function Timeline() {
  return (
    <div className="container" id="research">
      <h1 className="page-title">Research Papers</h1>

      <div className="timelineBody">
        <div className="timeline">
          {researchInfo.length > 0 &&
            researchInfo.map((ele) => (
              <TimelineResearch
                key={ele.slNo}
                cssClass={ele.cssClass}
                date={ele.date}
                title={ele.title}
                point1={ele.point1}
                point2={ele.point2}
                point3={ele.point3}
                number={ele.slNo}
```

```
                link={ele.link}
              />
            ))}
        </div>
      </div>
    </div>
  );
}

export default Timeline
```

3. TimelineResearch.js

```
import React from 'react'

function TimelineResearch(props) {
    return (
        <div>
            <div className={props.cssClass}>
                <div className="date">{props.date}</div>
                <div className="icon  pt-0" >
                    <div className='text-center' style={{ fontSize:
'21px', margin: '3px 0', fontWeight: 'bold' }}>{props.number}</div>
                </div>
                <div className="content">
                    <a href={props.link} target='_blank'><h2
className='timelineHeading link'>{props.title}</h2></a>
                    <ul className='bulletPoints'>
                        <li>{props.point1}</li>
                        <li>{props.point2}</li>
                        <li>{props.point3}</li>
                    </ul>
                </div>
            </div>
        </div>
    )}
export default TimelineResear
```

4. Timeline.css

```
.timelineBody,
.timelineBody::before,
.timelineBody::after {
  box-sizing: border-box;
```

```css
}

.timelineBody {
  margin: 0;
  font-family: Arial, Helvetica, sans-serif;
  background: #ffffff;
}

.timeline {
  position: relative;
  width: 100%;
  max-width: 1140px;
  margin: 0 auto;
  padding: 15px 0;
}

.timeline::after {
  content: "";
  position: absolute;
  width: 2px;
  background: rgb(12, 31, 152);
  top: 0;
  bottom: 0;
  left: 50%;
  margin-left: -1px;
}

.container4 {
  padding: 15px 30px;
  position: relative;
  background: inherit;
  width: 50%;
}

.container4.left {
  left: 0;
}

.container4.right {
  left: 50%;
}

.container4::after {
  content: "";
  position: absolute;
```

```
  width: 16px;
  height: 16px;
  top: calc(50% - 8px);
  right: -8px;
  background: #ffffff;
  border: 2px solid rgb(12, 31, 152);
  border-radius: 16px;
  z-index: 1;
}

.container4.right::after {
  left: -8px;
}

.container4::before {
  content: "";
  position: absolute;
  width: 50px;
  height: 2px;
  top: calc(50% - 1px);
  right: 8px;
  background: rgb(12, 31, 152);
  z-index: 1;
}

.container4.right::before {
  left: 8px;
}

.container4 .date {
  position: absolute;
  display: inline-block;
  top: calc(50% - 8px);
  text-align: center;
  font-size: 14px;
  font-weight: bold;
  color: rgb(12, 31, 152);
  text-transform: uppercase;
  letter-spacing: 1px;
  z-index: 1;
}

.container4.left .date {
  right: -95px;
}
```

```css
.container4.right .date {
  left: -95px;
}

.container4 .icon {
  position: absolute;
  display: inline-block;
  width: 40px;
  height: 40px;
  padding: 9px 0;
  top: calc(50% - 20px);
  background: linear-gradient(10deg, #540303 0%, #0d0473 100%);
  border: 2px solid rgb(12, 31, 152);
  border-radius: 40px;
  text-align: center;
  font-size: 18px;
  color: #d4e2df;
  z-index: 1;
}

.container4.left .icon {
  right: 56px;
}

.container4.right .icon {
  left: 56px;
}

.container4 .content {
  padding: 30px 90px 30px 30px;
  background: rgba(28, 160, 156, 0.286);
  position: relative;
  border-radius: 0 500px 500px 0;
}

.container4.right .content {
  padding: 30px 30px 30px 90px;
  border-radius: 500px 0 0 500px;
}

.container4 .content h2 {
  margin: 0 0 10px 0;
  font-size: 100%;
  font-weight: normal;
```

```
  color: #001953;
}

.container4 .content li {
  margin: 0;
  font-size: 90%;
  line-height: 19px;
  color: #2e2d2dae;
}

.timelineHeading {
  font-family: "Roboto", sans-serif;
}

ul.bulletPoints {
  list-style-type: square;
}

@media (max-width: 995px) {
  .timeline::after {
    left: 7px;
  }

  .container4 {
    width: 100%;
    padding-left: 5%;
    padding-right: 10px;
  }

  .container4.right {
    left: 0%;
  }

  .container4.left::after,
  .container4.right::after {
    left: 0px;
  }

  .container4.left::before,
  .container4.right::before {
    left: 0px;
    border-color: transparent rgb(12, 31, 152) transparent transparent;
  }

  .container4.left .date,
```

```
  .container4.right .date {
    right: 3%;
    left: auto;
    top: calc(78%);
    width: 100px;
  }

  .container4.left .icon,
  .container4.right .icon {
    right: auto;
    left: 7.1%;
  }

  .container4.left .content,
  .container4.right .content {
    padding: 30px 5% 40px 15%;
    border-radius: 300px 0 0 300px;
  }
  .container4 .content h2 {
    font-size: 100%;
  }

  .container4 .content li {
    font-size: 85%;
    line-height: 20px;
  }
}
@media (max-width: 450px) {
  .timeline::after {
    left: 7px;
  }

  .container4 {
    width: 100%;
    padding-left: 5%;
    padding-right: 10px;
  }

  .container4.right {
    left: 0%;
  }
  .container4.left::after,
  .container4.right::after {
    left: 0px;
  }
```

```
  .container4.left::before,
  .container4.right::before {
    left: 0px;
    border-color: transparent rgb(12, 31, 152) transparent transparent;
  }

  .container4.left .date,
  .container4.right .date {
    right: 3%;
    left: auto;
    top: calc(83%);
    width: 100px;
  }

  .container4.left .icon,
  .container4.right .icon {
    right: auto;
    left: 11.5%;
  }
  .container4.left .content,
  .container4.right .content {
    padding: 30px 5% 40px 18%;
    border-radius: 130px 0 0 130px;
  }
  .container4 .content h2 {
    font-size: 100%;
  }

  .container4 .content li {
    font-size: 85%;
    line-height: 20px;
  }
}

.link:hover {
  text-decoration: underline;
  font-style: italic;
```

Algorithm Steps Section:

1. **StepsContent.js**

```
const StepsContent = [
```

```
    {
        slNo: '1',
        heading: 'Importing Required Libraries',
        description: 'Before we begin, we require the following libraries
and dependencies, which needs to be imported into our Python environment
such as NumPy, TensorFlow, OS, Python, Pil, Brainweb, Tqdm, Logging,
Matplotlib, OpenCv2, Math, Skimage and random.'
    },
    {
        slNo: '2',
        heading: 'Collecting the Dataset',
        description: 'During this step, we fetch our MRI datasets (around
7200 images) from Brainweb, which are in grayscale and proceed with
importing images and exporting images to tiff format.'
    },
    {
        slNo: '3',
        heading: 'Dividing Dataset',
        description: 'The collected MRI datasets are segregated into
datasets of Original Image, Noisy Image & Denoised Image for the purpose
of Training and Testing in the upcoming steps.'
    },
    {
        slNo: '4',
        heading: 'Resizing the Image',
        description: 'During this step, the MRI datasets are converted to
NumPy form (array) and resized to 400, 400 pixels to uniformly size and
shape the images. This step is considered one of the important steps to
be followed and to improve the accuracy of the model during the training
phase.'
    },
    {
        slNo: '5',
        heading: 'Adding Noise to Dataset',
        description: 'Noise in the range of standard deviation 0-45 is
added to the images which are resized (400, 400 pixels). The noise is
based on the Rician noise formula as MRI images are mostly predominant
with Rician noise.'
    },
    {
        slNo: '6',
        heading: 'Creating CNN model',
        description: 'A CNN model is created with the help of encoders
and decoders. Using Maxpooling 2D and Convo 2D layer concepts, we create
Autoencoders and Decoders. Where in the model, Relu and Sigmoid are
```

```
implemented as activation functions. And also, Adam is used as an
optimizer for autoencoders.'
    },
    {
        slNo: '7',
        heading: 'Training the Model',
        description: 'Using the MRI datasets collected from Brainweb
which were segregated into datasets of Original image, Noisy image &
Denoised image is used for training the created CNN model.'
    },
    {
        slNo: '8',
        heading: 'Testing the Model',
        description: 'The model is tested by using testing dataset which
was collected during segregation of datasets. Extensive Empirical study
is performed to test and validate the data.'
    },
    {
        slNo: '9',
        heading: 'Modifying Model to Increase Accuracy',
        description: 'Once the model is validated for undesired value of
PSNR we procced with modifying the model to improve the accuracy rate
using Extensive Empirical study and proceed with training and testing the
model again till we obtain desired results.'
    },
    {
        slNo: '10',
        heading: 'Getting the Desired output',
        description: 'The last step includes capturing the desired output
value of PSNR, which is sufficient to denoise an image and excels over
other existing models and proceed building an application to denoise
medical images.This step also includes converting the NumPy images into
tiff image format and we try to save it to the application.'
    },
]

export default StepsContent
```

2. AlgorithmSteps.js

```
import React, { useState, useEffect } from 'react'
import Steps from './Steps'
import StepsContent from './StepsContent'
```

```
function AlgorithmSteps() {
    const [isDesktop, setDesktop] = useState(window.innerWidth >= 576);

    const updateMedia = () => {
        setDesktop(window.innerWidth >= 576);
    };

    useEffect(() => {
        window.addEventListener("resize", updateMedia);
        return () => window.removeEventListener("resize", updateMedia);
    });
    return (
        <div id='algo' className='mt-4 mb-5 navBarMargin'>
            <div className="container">

                <h1 className='page-title'>Algorithm Details</h1>
            </div>
            <div className='stepsContainer mt-4' >
                <div className="container pb-4">
                    <p className='mt-4'>Presenting a CNN-based algorithm
with the help of encoders and decoders using ReLu, Sigmoid as activation
function and Adam as an optimizer to find the optimum fit and to train
the efficient model using Extensive Empirical Study using data sets from
Brainweb. Estimating the original image by suppressing noise from a noise
contaminated version of the image by applying our efficient algorithm to
denoise the image with the highest SSIM and PSNR values with low
processing time.</p></div>

                {isDesktop ? <div className="row ">
                    {StepsContent.length > 0 && StepsContent.map((ele,
index) => (
                        <Steps number={ele.slNo} heading={ele.heading}
description={ele.description} key={ele.slNo} value={index} />
                    ))}
                </div>
                    :
                    <div className="row">
                        {StepsContent.length > 0 &&
StepsContent.map((ele, index) => (
                            <Steps number={ele.slNo}
heading={ele.heading} description={ele.description} key={ele.slNo}
value={index} />
                        ))}
```

```
                    </div>
                }



            </div>
        </div >
    )
}

export default AlgorithmSteps
```

3. Steps.js

```
import React, { useState, useEffect } from "react";

function Steps(props) {
    const [isDesktop, setDesktop] = useState(window.innerWidth >= 576);

    const updateMedia = () => {
        setDesktop(window.innerWidth >= 576);
    };

    useEffect(() => {
        window.addEventListener("resize", updateMedia);
        return () => window.removeEventListener("resize", updateMedia);
    });
    return (
        // Ternary Operator to display arrows and preventing it to
display at last
        <>
            <div className="col-xl-2 col-lg-2 col-md-2 col-sm-3 col-12" >
                <center>  <div className="v3_24 mt-0">
                    <p className="v3_28 text-center">{props.number}</p>
                </div></center>
                {/* Ternary Operator to check Screen size if less tham
576 add another arrow at last */}
                {isDesktop ?
                    // Ternary Operator to display arrows except the last
one
                    props.value < 9 ?
                        // 2nd Ternary True Condition
                        <center><div style={{ fontSize: '9vh',
fontWeight: 'bold', color: 'black' }} className='mb-0'>
&#x2193;</div></center> :
```

```
                        // 2nd Ternary False Condition
                        <div></div> :
                    // 1st Ternary False Condition -> Display arrows to
all
                    <center><div style={{ fontSize: '9vh', fontWeight:
'bold', color: 'black' }} className='mb-0'> &#x2193;</div></center>
                }




            </div>
            <div className="col-xl-10 col-lg-10 col-md-10 col-sm-9 col-12
mt-3">
                <h4 className='ml-0 timelineHeading mb-3
contentCenter'>{props.heading}</h4>
                <p > {props.description}</p>
            </div>
        </>
    )
}

export default Steps
```


4. Steps.css

```
.v3_28 {
  color: rgb(255, 255, 255);

  padding: 15px 0;
  font-family: Poppins;
  font-weight: Bold;
  font-size: 36px;
  opacity: 1;
}

.v3_24 {
  width: 85px;
  height: 85px;
  background: linear-gradient(10deg, #540303 0%, #0d0473 100%);
  opacity: 1;
  margin-top: 150px;
```

```
  border-radius: 50%;
}

.stepsContainer {
  padding: 0 10%;
}


.container2 {
  background-color: white;
  text-align: center;
  margin-bottom: 50px;
}

@media (max-width: 576px) {
  .contentCenter {
    text-align: center;
  }
}
```

Working Algorithm Section:

1. ImageComparison.js

```
import React, { useRef } from "react";
import ImageSlider from "react-image-comparison-slider";
import "./ImageComparison.css";
import ImageSliderCustom from "./ImageSliderCustom";
import ModalNav from "./ModalNav";

function ImageComparison() {
  const [modalShow, setModalShow] = React.useState(false);
  return (
    <div>
      <div className="container navBarMargin" id="working">
        <h1 className="mt-5 mb-5 page-title">Working of Algorithm</h1>
      </div>
      <div className="team pt-5">
        <div className="container3  container mt-3">
          <div className="row">
            <div className="col-xl-6 col-lg-6 col-md-12 col-sm-12 text-
center">
              <div className="img-fluid imageSli">
```

```
                <h4 style={{ fontFamily: "Roboto,sans-serif" }}>
                  SLIDE TO COMPARE RESULT
                </h4>
                <ImageSliderCustom
                  firstImage="images/output.png"
                  secondImage="images/input.png"
                />
              </div>
            </div>

            <div className="col-xl-6 col-lg-6 col-md-12 col-sm-12">
              <h4 className="mt-3 card-title text-center">
                Details About the Working
              </h4>
              <p className="mt-3 text-muted" style={{ padding: "0 5%" }}>
                Now let us see how the model is trained and its accuracy
when
                compared to other methods. In order to find their
accuracy, we
                have used PSNR values to compare. Firstly, the model
contains of
                ENCODERS, DECODERS and AUTOENCODERS.
              </p>
              <ul>
                <li>
                  In ENCODERS, we used convolution2D layers with
activation
                  function relu and padding as same. We have used 2
layers of
                  this layer along with MAX pooling 2D with padding as
same, and
                  N number of hidden layers with activation relu.
                </li>
                <li>
                  {" "}
                  In DECODING, we used convolution2D Transpose layers
with
                  activation relu and padding as same.{" "}
                </li>
                <li>
                  {" "}
                  In AUTOENCODERS, to compile we have used optimizer Adam
and
                  loss function as Binary_Crossentropy and the model has
given
```

```
                promising results.
              </li>
            </ul>{" "}
            Original PSNR Value of Noisy Image &#x2192; 22.21
            <br />
            <p>
              {" "}
              Our Model's Accuracy of Denoised Image gives PSNR Value
&#x2192;
              28.79
            </p>
            <div className="text-center">
              <button
                className="btn my-2 my-sm-0 mx-3 "
                type="submit"
                onClick={() => setModalShow(true)}
                style={{ background: "black", color: "white" }}
              >
                CLICK HERE TO UPLOAD YOUR IMAGE
              </button>
            </div>
            <ModalNav show={modalShow} onHide={() =>
setModalShow(false)} />
          </div>
          <div className="col-xl-3  col-lg-3 col-md-12 col-sm-
12"></div>
          <div className="col-xl-6  col-lg-6 col-md-12 col-sm-12">
            <h4 className="mt-2 card-title text-center">
              Acccuracy Rate of Models
            </h4>
            <table className="table my-4">
              <thead className="thead-dark">
                <tr>
                  <th scope="col">#</th>
                  <th scope="col">Filter / Method</th>
                  <th scope="col">PSNR Value</th>
                </tr>
              </thead>
              <tbody>
                <tr>
                  <th scope="row">1</th>
                  <td>Gaussian Filter</td>
                  <td>18.08</td>
                </tr>
                <tr>
```

```
                <th scope="row">2</th>
                <td>Denoise Bilateral</td>
                <td>18.22</td>
              </tr>
              <tr>
                <th scope="row">3</th>
                <td>Img_Aniso_Filter</td>
                <td>18.36</td>
              </tr>
              <tr>
                <th scope="row">4</th>
                <td>BM3D Denoise</td>
                <td>21.88</td>
              </tr>
              <tr>
                <th scope="row">5</th>
                <td>Denoise Wavelet</td>
                <td>22.43</td>
              </tr>
              <tr>
                <th scope="row">6</th>
                <td>Shift Inv Wavelet</td>
                <td>22.46</td>
              </tr>

              <tr>
                <th scope="row">7</th>
                <td>Denoise_Img_As_8byte</td>
                <td>22.63</td>
              </tr>

              <tr>
                <th scope="row">8</th>
                <td>Previous Model</td>
                <td>27.15</td>
              </tr>
              <tr>
                <th scope="row">9</th>
                <td>Current Model</td>
                <td>28.79</td>
              </tr>
            </tbody>
          </table>
        </div>
```

```
            <div className="col-xl-3  col-lg-3 col-md-12 col-sm-
12"></div>
          </div>
        </div>
      </div>
    </div>
  );
}

export default ImageComparison;
```

2. ImageSliderCustom.js

```
import React from 'react'

function ImageSliderCustom(props) {
    return (
        <div >
            <img-comparison-slider  >
                <img slot="first" src={props.firstImage} style={{
maxWidth: '400px', width: '100%', height: 'auto' }} />
                <img slot="second" src={props.secondImage} style={{
maxWidth: '400px', width: '100%', height: 'auto' }} />
            </img-comparison-slider>
        </div>
    )
}

export default ImageSliderCustom
```

3. ImageComparison.css

```
.imageSli {
  width: "65%";
  height: "50vh";
  margin-top: "5vh";
  margin-bottom: "5vh";
  margin-left: "auto";
  margin-right: "auto";
}

img-comparison-slider {
  --divider-width: 3px;
```

```
  --divider-color: red;
  --default-handle-color: rgb(255, 5, 5);
}

.navBarMargin::before {
  display: block;
  content: " ";
  margin-top: -43px;
  height: 43px;
  visibility: hidden;
  pointer-events: none;
}
```

Tools Section:

1. **ToolsUsed.js**

```
import React from "react";
import Slider from "react-slick";

function ToolsUsed() {
  var settings = {
    // dots: true,
    autoplaySpeed: 1850,
    autoplay: true,
    slidesToShow: 6,
    slidesToScroll: 5,
    initialSlide: 0,
    centerPadding: "200px",
    accessibility: true,
    rows: "1",

    responsive: [
      {
        breakpoint: 1920,
        settings: {
          slidesToShow: 6,
          slidesToScroll: 5,
          infinite: true,
          lazyLoad: "ondemand",
          dots: true,
        },
      },
      {
```

```
        breakpoint: 1390,
        settings: {
          slidesToShow: 5,
          slidesToScroll: 4,
          infinite: true,
          lazyLoad: "ondemand",
          dots: true,
        },
      },
      {
        breakpoint: 1024,
        settings: {
          slidesToShow: 4,
          slidesToScroll: 3,
          infinite: true,
          lazyLoad: "ondemand",
          dots: true,
        },
      },
      {
        breakpoint: 600,
        settings: {
          slidesToShow: 3,
          slidesToScroll: 2,
          infinite: true,
          lazyLoad: "ondemand",
          dots: true,
        },
      },
      {
        breakpoint: 480,
        settings: {
          slidesToShow: 2,
          slidesToScroll: 2,
          infinite: true,
          lazyLoad: "ondemand",
          dots: true,
        },
      },
    ],
  };
  return (
    <div className="container mb-5 navBarMargin" id="tools">
      <h1 className="page-title">Tools / Libraries / Frameworks</h1>
      <div className="container">
```

```
        <Slider {...settings}>
          <img src="images/python.png" alt="Python logo" />
          <img src="images/tensorflow.png" alt="Tensorflow logo" />
          <img src="images/matplotlib.jpg" alt="Matplotlib logo" />
          <img src="images/scikit.jpg" alt="Scikit logo" />
          <img src="images/keras.jpg" alt="Keras logo" />
          <img src="images/kaggle.png" alt="Kaggle logo" />
          <img src="images/reactjs.png" alt="ReactJs logo" />
          <img src="images/bootstrap.png" alt="Boootsrap logo" />
          <img src="images/javascript2.jpg" alt="Javascript logo" />
          <img src="images/jquery.png" alt="Jquery logo" />
          <img src="images/aws.png" alt="AWS logo" />
          <img src="images/flask.png" alt="Flask logo" />
        </Slider>
      </div>
    </div>
  );
}

export default ToolsUsed;
```

2. **Tools.css**

```
@import
url("https://fonts.googleapis.com/css2?family=Poppins&display=swap");
@media (max-width: 600px) {
  html,
  body {
    overflow-x: hidden;
  }}
```

Teams Section

1. **TeamDetails.js**

```
const teamDetails = [
  {
    image: "images/yamanappa.jpg",
    personName: "Prof Yamanappa",
    title: "Assistant Professor - CSE",
    content: "Area of Research : Medical Image Ananlysis, Computer Vision
",
```

```
    linkedin: "",
    email: "",
    github: "",
    instagram: "",
  },

  {
    image: "images/abdullah.jpg",
    personName: "Mohammed Abdullah",
    title: "20181CSE0426",
    content:
      "Full Stack Developer along with core image processing Developed
CNN model along with data filtering and data processing,  noise creation
and adding noise, and trained the CNN model.",
    linkedin: "https://www.linkedin.com/in/mohammed-abdullah-6268611b9/",
    email: "mohammedabdullah0212@gmail.com",
    github: "https://github.com/abdullah7795",
    instagram: "https://www.instagram.com/_abdullah_02__/",
  },
  {
    image: "images/saif.jpg",
    personName: "Mohammed Saif",
    title: "20181CSE0433",
    content:
      "React JS Developer & UI/UX designing. As a Team Leader developing
a strategic & systematic plan to accomplish predetermined tasks and
ensuring that there is a collaborative work environment.",
    linkedin: "https://www.linkedin.com/in/mohammedsaif001/",
    email: "mdsaif4online@gmail.com",
    github: "https://github.com/mohammedsaif001",
    instagram: "https://www.instagram.com/mohammedsaif001/",
  },
  {
    image: "images/nishithaa_photo.jpg",
    personName: "Nishitaa Palani",
    title: "20181CSE0490",
    content:
      "Development of Image Denoising algorithm and CNN model,Training of
model, Creation of presentation,documentation and website content.",
    linkedin: "https://www.linkedin.com/in/nishithaa-palani/",
    email: "nishithaapalani@gmail.com ",
    github: "https://github.com/NishithaaPalani/NP.git",
    instagram: "https://www.instagram.com/nishithaa_palani/",
  },
  {
```

```
    image: "images/yunus.jpeg",
    personName: "Syed Yunus",
    title: "20181CSE0737 ",
    content:
      "Backend Developer, created a backend on which the image processing
model is embedded using flask and also  created android application.",
    linkedin: "https://www.linkedin.com/in/syed-yunus-b677531a9",
    email: "syedyunusstar@gmail.com ",
    github: "https://github.com/wideflare/",
    instagram: "https://www.instagram.com/syedyunusstar/",
  },

  {
    image: "images/uvais.jpg",
    personName: "Mohammed Uvais",
    title: "20181CSE0437",
    content:
      "UI/UX designing ,Content Creation,Documentation and Website
Testing ",
    linkedin: "https://www.linkedin.com/in/mohammeduvais/",
    email: "mohammeduvais.premium@gmail.com",
    github: "https://github.com/UvaisPremium",
    instagram: "https://www.instagram.com/mohammed_uvais_premium/?hl=en",
  },
];

export default teamDetails;
```


2. Team.js

```
import React from 'react'

import teamDetails from './TeamDetails'
import TeamCard from './TeamCard'
function Team() {
    return (
        <>
            <div className="container navBarMargin">
                <h1 className="page-title">
                    Team Member Details
                </h1>
            </div>
            <div className='team ' id='aboutus'>
```

```
                <div className='container1 mx-auto mt-5 col-md-10 mt-
100'>
                    <div className="header">
                        <div className="title">Presidency University
Bangalore </div>
                        <p><small className="text-muted">Computer Science
Engineering (2018 - 2022) </small></p>

                    </div>
                    <div className="row justify-content-center pb-5
individualCard" >
                        {teamDetails.length > 0 &&
teamDetails.map((element) => (

                            <TeamCard image={element.image}
personName={element.personName} title={element.title}
content={element.content} linkedin={element.linkedin}
email={element.email} github={element.github}
instagram={element.instagram} key={element.personName} />

                        ))}

                    </div>
                </div>

            </div>
        </>
    )
}
export default Tea
```

3. TeamCard.js

```
import React from "react";
function TeamCard(props) {
    return (
        <>
            <div className="card card-ext col-md-3 mt-100"
key={props.personName}>
                <div className="card-content">
                    <div className="card-body card-body-ext p-0">
                        <div className="profile">
                            {" "}
                            <img src={props.image} />{" "}
                        </div>
```

```
                        <div className="card-title mt-4">
                            {" "}
                            {props.personName}
                            <br /> <small>{props.title}</small>{" "}
                        </div>
                        <div className="card-subtitle">
                            <p>
                                {" "}
                                <small className="text-muted">
{props.content} </small>{" "}
                            </p>
                            <ul className="social mb-0 list-inline mt-3">
                                <li className="list-inline-item">
                                    <a target="_blank"
href={props.linkedin} className="social-link">
                                        <i className="fa fa-
linkedin"></i>
                                    </a>
                                </li>
                                <li className="list-inline-item">
                                    <a target="_blank"
href={`mailto:${props.email}`} className="social-link">
                                        <i className="fa fa-
envelope"></i>
                                    </a>
                                </li>
                                <li className="list-inline-item">
                                    <a target="_blank"
href={props.github} className="social-link">
                                        <i className="fa fa-github"></i>
                                    </a>
                                </li>
                                <li className="list-inline-item">
                                    <a target="_blank"
href={props.instagram} className="social-link">
                                        <i className="fa fa-
instagram"></i>
                                    </a>
                                </li>
                            </ul>
                        </div>
                    </div>
                </div>
            </div>
        </>
```

```
    );
}

export default TeamCard;
```

4. Team.css

```
.team {
  background: linear-gradient(90deg, #540303 0%, #000000 100%);
  padding-top: 10px;
  padding-bottom: 5px;
}

.container1 {
  background-color: white;
  text-align: center;
  border-radius: 20px;
  box-shadow: 0 20px 40px rgba(0, 0, 0, 0.2);
  margin-bottom: 50px;
}

.container3 {
  background-color: rgb(244, 244, 244);
  /* text-align: center; */
  border-radius: 20px;
  box-shadow: 0 20px 80px rgba(0, 0, 0, 0.2);
  margin-bottom: 50px;
  padding-bottom: 40px;
  padding-top: 40px;
}

.title {
  font-size: 25px;
  font-weight: 100;
}

.icon {
  position: relative;
  bottom: 11px;
}

.mt-100 {
  margin-top: 50px;
}
```

```css
.profile img {
  width: 68px;
  height: 68px;
  border-radius: 50%;
}

.card-ext {
  border-radius: 15px;
  margin-left: 30px;
  margin-right: 30px;
  box-shadow: 0 10px 20px rgba(0, 0, 0, 0.2);
}

.card-body-ext {
  position: relative;
  bottom: 35px;
}

.header {
  padding-top: 40px;
}

.social-link {
  width: 30px;
  height: 30px;
  border: 1px solid #ddd;
  display: flex;
  align-items: center;
  justify-content: center;
  color: #666;
  border-radius: 50%;
  transition: all 0.3s;
  font-size: 0.9rem;
  text-decoration: none;
}

.social-link:hover,
.social-link:focus {
  background: #ddd;
  text-decoration: none;
  color: #555;
}

.dragDrop {
```

```
  cursor: pointer;
}

.card {
  height: 100%;
}
```

FAQ's Section

1. **AccordionQuestions.css**

```
.accordion .card {
  background: none;
  border: none;
}
.accordion .card .card-header {
  background: none;
  border: none;
  padding: 0.4rem 1rem;
  font-family: "Roboto", sans-serif;
}
.accordion .card-header h2 span {
  float: left;
  margin-top: 10px;
}
.accordion .card-header .btn {
  color: #2f2f31;
  font-size: 1.04rem;
  text-align: left;
  position: relative;
  font-weight: 500;
  padding-left: 2rem;
}
.accordion .card-header i {
  font-size: 1.2rem;
  font-weight: bold;
  position: absolute;
  left: 0;
  top: 9px;
}
.accordion .card-header .btn:hover {
  color: #73bb2b;
}
.accordion .card-body {
```

```
  color: #324353;
  padding: 0.5rem 3rem;
}
.page-title {
  margin: 3rem 0 3rem 1rem;
  font-family: "Roboto", sans-serif;
  position: relative;
}
.page-title::after {
  content: "";
  width: 80px;
  position: absolute;
  height: 3px;
  border-radius: 1px;
  background: #73bb2b;
  left: 0;
  bottom: -15px;
}

.accordion .highlight .btn {
  color: #ff0000;
}
.accordion .highlight i {
  transform: rotate(180deg);
}
```


2. AccordionQuestions.js

```
import React, { Component } from "react";
import $ from "jquery";
export default class AccordionQuestions extends Component {
  componentDidMount = () => {
    $(".collapse.show").each(function () {
      $(this).prev(".card-header").addClass("highlight");
    });

    // Highlight open collapsed element
    $(".card-header .btn").on("click", function () {
      $(".card-header").not($(this).parents()).removeClass("highlight");
      $(this).parents(".card-header").toggleClass("highlight");
    });
  };
```

```
  render() {
    return (
      <div className="container-lg" id="faq">
        <div className="row">
          <div className="col-lg-12">
            <h1 className="page-title">Frequently Asked Questions</h1>
            <div className="accordion" id="accordionExample">
              <div className="card">
                <div className="card-header" id="headingOne">
                  <h2 className="clearfix mb-0">
                    <a
                      className="btn btn-link"
                      data-toggle="collapse"
                      data-target="#collapseOne"
                      aria-expanded="true"
                      aria-controls="collapseOne"
                    >
                      <i className="fa fa-chevron-circle-down"></i> Is
this
                      Algorithm Applicable only for Medical Images?
                    </a>
                  </h2>
                </div>
                <div
                  id="collapseOne"
                  className="collapse show"
                  aria-labelledby="headingOne"
                  data-parent="#accordionExample"
                >
                  <div className="card-body">
                    Yes, This Algorithm is Applicable only for Medical
Images.
                    The reason being, our Model is trained to Remove
Ration
                    Noise Which is Present Mainly in MRI Images and hence
our
                    Model is Limited only to Medical Images .{" "}
                  </div>
                </div>
              </div>
              <div className="card">
                <div className="card-header" id="headingTwo">
                  <h2 className="mb-0">
                    <a
                      className="btn btn-link collapsed"
```

```
                    data-toggle="collapse"
                    data-target="#collapseTwo"
                    aria-expanded="false"
                    aria-controls="collapseTwo"
                  >
                    <i className="fa fa-chevron-circle-down"></i> Why
Can't I
                    Upload Regular Images to Denoise?
                  </a>
                </h2>
              </div>
              <div
                id="collapseTwo"
                className="collapse"
                aria-labelledby="headingTwo"
                data-parent="#accordionExample"
              >
                <div className="card-body">
                  The Regular Images or Day-to-Day Images Contains
Different
                  Noises namely Gaussian Noise, Salt &amp; Pepper
Noise,
                  Speckle Noise, Poisson Noise, Impulse Noise and much
more
                  but mostly not Ration noise. Whereas our Model is
solely to
                  denoise Ration Noise which is more applicable to
Medical
                  Images.
                </div>
              </div>
            </div>
            <div className="card">
              <div className="card-header" id="headingThree">
                <h2 className="mb-0">
                  <a
                    className="btn btn-link collapsed"
                    data-toggle="collapse"
                    data-target="#collapseThree"
                    aria-expanded="false"
                    aria-controls="collapseThree"
                  >
                    <i className="fa fa-chevron-circle-down"></i> Why
am I
                    Unable to Upload Images in PNG/JPG/JPEG/SVG/GIF?
```

```
                    </a>
                  </h2>
                </div>
                <div
                  id="collapseThree"
                  className="collapse"
                  aria-labelledby="headingThree"
                  data-parent="#accordionExample"
                >
                  <div className="card-body">
                    Our Model is Developed Stricly to Denoise Medical
Images
                    &amp; Since Most MRI Images Format is TIFF/TIF and
Not
                    PNG/JPG/JPEG/SVG therefore our Model Doesn't Allow
User to
                    Upload These Format Images so that it Doesn't Spoil
the
                    Uploaded image.
                  </div>
                </div>
              </div>
              <div className="card">
                <div className="card-header" id="headingFour">
                  <h2 className="mb-0">
                    <a
                      className="btn btn-link collapsed"
                      data-toggle="collapse"
                      data-target="#collapseFour"
                      aria-expanded="false"
                      aria-controls="collapseFour"
                    >
                      <i className="fa fa-chevron-circle-down"></i> Why
this
                      Algorithm is Working only for Black &amp; White or
                      Grayscale images?{" "}
                    </a>
                  </h2>
                </div>
                <div
                  id="collapseFour"
                  className="collapse"
                  aria-labelledby="headingFour"
                  data-parent="#accordionExample"
                >
```

```
                  <div className="card-body">
                    Since MRI Images are Captured in Grayscale Format and
the
                    Hard Copy of the Image is in Black and White; It's
More
                    Efficient to Output the Model in Grayscale Image Than
RGB
                    Values.
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>
      );
  }
}
```

UploadImage.js

```
import React, { useState } from "react";
import ImageSliderCustom from "./ImageSliderCustom";

import { Alert } from "react-bootstrap";

// Import React FilePond
import { FilePond, registerPlugin } from "react-filepond";

// Import FilePond styles
import "filepond/dist/filepond.min.css";

// Import the Image EXIF Orientation and Image Preview plugins
// Note: These need to be installed separately
// `npm i filepond-plugin-image-preview filepond-plugin-image-exif-
orientation --save`
import FilePondPluginImageExifOrientation from "filepond-plugin-image-
exif-orientation";
import FilePondPluginImagePreview from "filepond-plugin-image-preview";
import "filepond-plugin-image-preview/dist/filepond-plugin-image-
preview.css";

// Register the plugins
```

```
registerPlugin(FilePondPluginImageExifOrientation,
FilePondPluginImagePreview);

function UploadImage() {
  const [image, setImage] = useState(null);
  const [files, setFiles] = useState([]);
  const [error, setError] = useState(false);
  const [serverIssue, setServerIssue] = useState(false);
  const [serverOverload, setServerOverload] = useState(false);
  const [show, setShow] = useState(true);
  const [downloadImageStatus, setDownloadImageStatus] = useState(false);
  const [downloadImageUrl, setDownloadImageUrl] = useState(null);

  return (
    <div>
      <FilePond
        files={files}
        onupdatefiles={setFiles}
        allowMultiple={false}
        allowDrop={true}
        allowBrowse={true}
        maxFiles={1}
        server={{
          revert: (uniqueFileId, load, error) => {
            // Should remove the earlier created temp file here
            // ...

            console.log("fileid: " + uniqueFileId);
            // Can call the error method if something is wrong, should
exit after
            error("oh my goodness");

            // Should call the load method when done, no parameters
required
            load();
          },

          process: (
            fieldName,
            file,
            metadata,
            load,
            error,
            progress,
            abort,
```

```
            transfer,
            options
          ) => {
            const formData = new FormData();

            formData.append("image", file);

            const request = new XMLHttpRequest();
            request.open("POST", "https://172.105.52.26:5000");

            // Should call the progress method to update the progress to
100% before calling load
            // Setting computable to false switches the loading indicator
to infinite mode
            request.upload.onprogress = (e) => {
              progress(e.lengthComputable, e.loaded, e.total);
            };

            // Should call the load method when done and pass the
returned server file id
            // this server file id is then used later on when reverting
or restoring a file
            // so your server knows which file to return without exposing
that info to the client
            request.onload = function () {
              if (request.status >= 200 && request.status < 300) {
                // the load method accepts either a string (id) or an
object
                console.log(request.status);
                const obj = JSON.parse(request.responseText);
                console.log(obj);

                switch (obj.response.code) {
                  case 200:
                    load(obj.processedImage);
                    setDownloadImageStatus(true);
                    setDownloadImageUrl(obj.processedImage);
                    console.log(file);
                    setShow(true);
                    setImage(obj.processedImage);

                    break;

                  case 400:
                    switch (obj.response.status) {
```

```
                    case "image-format-invalid":
                      //alert user
                      setError(true);
                      setDownloadImageStatus(false);
                      setShow(true);
                      error("oh no");
                      break;
                  }

                  break;
              }

              //console.log(obj.imageUrl)
              //   setFiles([...files , file]);
            } else {
              // Can call the error method if something is wrong,
should exit after
              error("oh no");
              setShow(true);
              setServerIssue(true);
            }
          };

          request.send(formData);

          // Should expose an abort method so the request can be
cancelled
          return {
            abort: () => {
              // This function is entered if the user has tapped the
cancel button
              request.abort();

              // Let FilePond know the request has been cancelled
              abort();
            },
          };
        },
      }}
      instantUpload={true}
      name="files"
      labelIdle='Drag & Drop your files or <u><span
className="filepond--label-action dragDrop">Browse</span></u>'
    />
```

```
      {serverIssue ? (
        // If server is down or not active
        <Alert variant="warning" show={show} onClose={() =>
setShow(false)}>
          <Alert.Heading>We'll be back shortly :(</Alert.Heading>
          <p>
            Server is Unavailable at the Moment as it is Under
Maintenance.
            Please Try Again Later.
          </p>
        </Alert>
      ) : serverOverload ? (
        <Alert variant="warning" show={show} onClose={() =>
setShow(false)}>
          <Alert.Heading>Too Many Requests :(</Alert.Heading>
          <p>Sorry for the Inconvinience. Please Try Again Later.</p>
        </Alert>
      ) : downloadImageStatus ? (
        // Alert Dialog Box when Success & Option for downloading Image

        <Alert variant="success" show={show}>
          <Alert.Heading>Denoised Successfully :)</Alert.Heading>
          <p>
            Please Download Your Processed Image by Clicking on the
Button Given
            Below. Thank You!
          </p>
          <hr />
          <div className="text-center">
            <button
              className="btn btn-outline-success my-2 my-sm-0 mx-2"
              onClick={() => (window.location = downloadImageUrl)}
              type="submit"
            >
              DOWNLOAD YOUR IMAGE
            </button>
            <button
              className="btn btn-outline-danger my-2 my-sm-0 mx-2"
              onClick={() => {
                setError(false);
                setShow(false);
                setFiles([]);
              }}
              type="submit"
            >
```

```
              UPLOAD AGAIN
            </button>
          </div>
        </Alert>
      ) : error ? (
        // Alert Dialog Box when Error Occurs
        <Alert
          variant="danger"
          show={show}
          onClose={() => setShow(false)}
          dismissible
        >
          <Alert.Heading>Oh snap! You got an Error :(</Alert.Heading>
          <p>Please Upload an Image in TIF/TIFF format.</p>
          <div className="text-center">
            <button
              className="btn btn-outline-danger my-2 my-sm-0 mx-2"
              onClick={() => {
                setError(false);
                setShow(false);
                setFiles([]);
              }}
              type="submit"
            >
              UPLOAD AGAIN
            </button>
          </div>
        </Alert>
      ) : (
        <div></div>
      )}
    </div>
  );
}

export default UploadImage;
```

5.5 Backend Code

Code for web backend application using FLASK:

```
from flask import Flask  , request , jsonify , make_response import numpy as np
import tensorflow as tf import matplotlib.pyplot as plt import os

from tensorflow.keras import layers from
tensorflow.keras.datasets import mnist
from tensorflow.keras.models import
Model from IPython.display import
Image import imageio from PIL import
Image import time import os from os
import listdir import hashlib from
skimage import io import pickle import
json import boto3

from botocore.exceptions import ClientError
import pathlib

from flask_cors import CORS, cross_origin

#from .cv2 import *
#import cv2 from PIL
import Image app =
Flask(__name__)

#app.run(host="0.0.0.0",
port=8080) app.debug = True cors
= CORS(app)

app.config['CORS_HEADERS'] = 'Content-Type'

def upload_file(file_name, bucket, object_name=None):
"""Upload a file to an S3 bucket

:param file_name: File to upload

:param bucket: Bucket to upload to

:param object_name: S3 object name. If not specified then file_name is used
```

```
:return: True if file was uploaded, else False
"""

# If S3 object_name was not specified, use
file_name     if object_name is None:
object_name = file_name

# Upload the file

#s3_client = boto3.client('s3')

s3_client = boto3.client(service_name='s3',

endpoint_url = 'https://s3.filebase.com',
aws_access_key_id='E2C3DB6EE2C65941FD1A',

aws_secret_access_key='7vQg5qATii5tH6vcHMq7EeXWpgeRXGMO6jvOKyoy')
try:

response = s3_client.upload_file(file_name, bucket, object_name)
except ClientError as e:

logging.error(e)
return False     return
True def
preprocess1(array):


array = array.astype("float32") / 255.0
array = np.reshape(array, (1, 400, 400, 1))
return array

@app.route('/'  , methods = ['POST' , 'GET'])
def home():

if request.method == 'GET':                    json_data =   '{ "response": { "status":
"ready" , "code": 200  } , "universityName": "Presidency University" ,
"projectName": "Medical image denoising using autoencoders, CNN and deep
learning", "credits": [{ "studentName": "Syed Yunus" }  ,  {  "studentName":
"Mohammed Saif" } , { "studentName": "Mohammed Abdullah"  } , {
"studentName": "Nishita

Palani"  } , { "studentName": "Mohammed Uvais"   }  ] }'                    r =
make_response( json_data )                    r.mimetype = 'application/json'
return r
```

```
exit()
if request.method != 'POST':
json_data =  '{ "response": { "status": "bad-request" , "code": 400  } }'
r = make_response( json_data )                    r.mimetype = 'application/json'
return r
exit()
if 'image' in request.files:              pass         else:
json_data =  '{ "response": { "status": "image-not-set" , "code": 400  } }'
r = make_response( json_data )                    r.mimetype = 'application/json'
return r                    exit()
image = None
if request.files['image'].filename == '':
json_data =  '{ "response": { "status": "image-not-set" , "code": 400  } }'
r = make_response( json_data )                    r.mimetype = 'application/json'
return r
exit()
image = request.files['image']
if not ( image.filename.lower().endswith(('.tiff'))):

json_data =  '{ "response": { "status": "image-format-invalid" , "code": 400  } }'
r = make_response( json_data )                    r.mimetype = 'application/json'
return r                    exit()        ts = time.time()
result = hashlib.md5(str(ts).encode())         name = result.hexdigest()
new_model = tf.keras.models.load_model('saved_model/my_model')
autoencoder = new_model

#input1 = str(input("enter the file name dfjdnjdfnj  "))
# input1 = "T1.tif"
print(image)
#img = Image.open(image)         img = Image.open(image).convert('L')
print(img)
#       np_img1 = np.array(img)
# img = cv2.imread(img,0)         np_img1 = np.array(img)
print(np_img1.shape)
#   np_img1 = np.reshape(np_img1 , (400,400 ))
np_img1=preprocess1(np_img1)         print(np_img1.shape)
predictions2 = autoencoder.predict(np_img1)
for i in predictions2:              im =i
i = np.reshape(i,(400,400))
plt.imsave(name+'.tiff', i,cmap = 'gray')
```

```
#  print(im.shape)
upload_file(name+'.tiff', 'presidencyuniversity53','images/'+name+'.tiff')
json_data = ' { "response": { "status": "success" , "code": 200 } ,
"processedImage":
"'+'https://presidencyuniversity53.s3.filebase.com/images/'+name+'.tiff" } '
pathlib.Path(name+'.tiff').unlink()        json_object = json.loads(json_data)
r = make_response( json_data )        r.mimetype = 'application/json'        return
r
if __name__ == "__main__":
app.run(host = '0.0.0.0'  , ssl_context=('/etc/ssl/cert.crt', '/etc/ssl/private.key'))
```

Code for Android app :

MainActivity.java

```
package in.presidencyuniversity.imagedenoiseapp;

import androidx.appcompat.app.AppCompatActivity;

import android.content.ActivityNotFoundException;

import android.content.Intent; import

android.net.Uri; import android.os.Build; import

android.os.Bundle; import

android.webkit.ValueCallback; import

android.webkit.WebChromeClient; import

android.webkit.WebSettings; import

android.webkit.WebView; import

android.widget.Toast;

public class MainActivity extends AppCompatActivity {

   static WebView wv;
```

```
   private ValueCallback<Uri> mUploadMessage;
public ValueCallback<Uri[]> uploadMessage;
public static final int REQUEST_SELECT_FILE =
100;     private final static int
FILECHOOSER_RESULTCODE = 1;

   @Override

   protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);

        wv =  findViewById(R.id.wv);

       wv.loadUrl("https://mohammedsaif001.github.io/Medical-Image-Denoising/");

       WebSettings mWebSettings = wv.getSettings();
mWebSettings.setJavaScriptEnabled(true);
mWebSettings.setSupportZoom(false);
mWebSettings.setAllowFileAccess(true);
mWebSettings.setAllowContentAccess(true);


       wv.setWebChromeClient(new WebChromeClient()
       {
          // For 3.0+ Devices (Start)
          // onActivityResult attached before constructor
          protected void openFileChooser(ValueCallback uploadMsg, String acceptType)
          {
             mUploadMessage = uploadMsg;
             Intent i = new Intent(Intent.ACTION_GET_CONTENT);
i.addCategory(Intent.CATEGORY_OPENABLE);
             i.setType("image/*");
             startActivityForResult(Intent.createChooser(i, "File Browser"),
FILECHOOSER_RESULTCODE);
```

```
        }
        // For Lollipop 5.0+ Devices
        public boolean onShowFileChooser(WebView mWebView,
ValueCallback<Uri[]> filePathCallback, WebChromeClient.FileChooserParams
fileChooserParams)
        {
           if (uploadMessage != null) {
              uploadMessage.onReceiveValue(null);
              uploadMessage = null;}
           uploadMessage = filePathCallback;
           Intent intent =
fileChooserParams.createIntent();              try
           {
              startActivityForResult(intent, REQUEST_SELECT_FILE);
           } catch (ActivityNotFoundException e)
           {
              uploadMessage = null;
              Toast.makeText(MainActivity.this, "Cannot Open File Chooser",
Toast.LENGTH_LONG).show();
              return false;
           }
           return true;
}
        //For Android 4.1 only
        protected void openFileChooser(ValueCallback<Uri> uploadMsg, String
acceptType, String capture)
        {
```

```
            mUploadMessage = uploadMsg;
            Intent intent = new Intent(Intent.ACTION_GET_CONTENT);
            intent.addCategory(Intent.CATEGORY_OPENABLE);
intent.setType("image/*");
            startActivityForResult(Intent.createChooser(intent, "File Browser"),
FILECHOOSER_RESULTCO
DE);          }
        protected void openFileChooser(ValueCallback<Uri> uploadMsg)
        {
            mUploadMessage = uploadMsg;
            Intent i = new Intent(Intent.ACTION_GET_CONTENT);
i.addCategory(Intent.CATEGORY_OPENABLE);
            i.setType("image/*");
            startActivityForResult(Intent.createChooser(i, "File Chooser"),
FILECHOOSER_RESULTCODE);
        }
    });
}


   @Override
   public void onActivityResult(int requestCode, int resultCode, Intent intent)
   {
      super.onActivityResult(resultCode ,resultCode , intent );
      if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP)
      {
         if (requestCode == REQUEST_SELECT_FILE)
         {
```

```
            if (uploadMessage == null)
                return;

uploadMessage.onReceiveValue(WebChromeClient.FileChooserParams.parseResult(res
ultCode, intent));
            uploadMessage = null;
        }
    }
    else if (requestCode == FILECHOOSER_RESULTCODE)
    {
        if (null == mUploadMessage)
            return;
        // Use MainActivity.RESULT_OK if you're implementing WebView inside
Fragment
        // Use RESULT_OK only if you're implementing WebView inside an Activity
        Uri result = intent == null || resultCode != MainActivity.RESULT_OK ? null :
intent.getData();           mUploadMessage.onReceiveValue(result);
mUploadMessage = null;
    }
else
        Toast.makeText(MainActivity.this, "Failed to Upload Image",
Toast.LENGTH_LONG).show();
   }
}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">

   <WebView

      android:layout_width="match_parent"
android:layout_height="match_parent"
android:id="@+id/wv"  />

</RelativeLayout>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
package="in.presidencyuniversity.imagedenoiseapp">

  <uses-permission android:name="android.permission.INTERNET"></uses-
permission>

   <application

      android:allowBackup="true"

      android:dataExtractionRules="@xml/data_extraction_rules"
android:fullBackupContent="@xml/backup_rules"
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:roundIcon="@mipmap/ic_launcher"
android:supportsRtl="true"

      android:theme="@style/Theme.AppCompat.DayNight.NoActionBar"
tools:targetApi="31">

      <activity
```

```
        android:name=".MainActivity"
android:exported="true">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
      </activity>
   </application>
</manifest>
```

6. Training and Testing of Model

6.1 Training

```
autoencoder.fit
(
   x=train_data,
   y=train_data,
   epochs=50,
   batch_size=20,
   shuffle=True,
   validation_data= (test_data, test_data),
)

autoencoder. save('saved_model/my_model5')
```

```
autoencoder.fit
(
   x=noisy_train_data,
   y=train_data,
   epochs=100,
   batch_size=20,
   shuffle=True,
   validation_data= (noisy_test_data, test_data),
)

autoencoder. save('saved_model/my_model5_1')
```

In [10]:
```
autoencoder.fit(
    x=train_data,
    y=train_data,
    epochs=50,
    batch_size=20,
    shuffle=True,
    validation_data=(test_data, test_data),
)
autoencoder.save('saved_model/my_model5')
autoencoder.fit(
    x=noisy_train_data,
    y=train_data,
    epochs=100,
    batch_size=20,
    shuffle=True,
    validation_data=(noisy_test_data, test_data),
)
autoencoder.save('saved_model/my_model5_1')
```

```
254/254 [==============================] - 880s 3s/step - loss: 0.3033 - val_loss: 0.2956
Epoch 9/50
254/254 [==============================] - 967s 4s/step - loss: 0.3031 - val_loss: 0.2953
Epoch 10/50
254/254 [==============================] - 986s 4s/step - loss: 0.3029 - val_loss: 0.2951
Epoch 11/50
254/254 [==============================] - 978s 4s/step - loss: 0.3028 - val_loss: 0.2949
Epoch 12/50
254/254 [==============================] - 960s 4s/step - loss: 0.3026 - val_loss: 0.2948
Epoch 13/50
254/254 [==============================] - 941s 4s/step - loss: 0.3024 - val_loss: 0.2947
```

```
254/254 [==============================] - 886s 3s/step - loss: 0.3006 - val_loss: 0.2929
Epoch 46/50
254/254 [==============================] - 887s 3s/step - loss: 0.3007 - val_loss: 0.2927
Epoch 47/50
254/254 [==============================] - 886s 3s/step - loss: 0.3005 - val_loss: 0.2928
Epoch 48/50
254/254 [==============================] - 894s 4s/step - loss: 0.3005 - val_loss: 0.2927
Epoch 49/50
254/254 [==============================] - 909s 4s/step - loss: 0.3006 - val_loss: 0.2926
Epoch 50/50
254/254 [==============================] - 903s 4s/step - loss: 0.3005 - val_loss: 0.2926
```

```
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_con
lution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing 5
be directly callable after loading.
```

```
INFO:tensorflow:Assets written to: saved_model/my_model5\assets
```

```
INFO:tensorflow:Assets written to: saved_model/my_model5\assets
```

```
    batch_size=20,
    shuffle=True,
    validation_data=(test_data, test_data),
)
autoencoder.save('saved_model/my_model5')
autoencoder.fit(
    x=noisy_train_data,
    y=train_data,
    epochs=100,
    batch_size=20,
    shuffle=True,
    validation_data=(noisy_test_data, test_data),
)
autoencoder.save('saved_model/my_model5_1')
```

```
Epoch 95/100
254/254 [==============================] - 962s 4s/step - loss: 0.3056 - val_loss: 0.2984
Epoch 96/100
254/254 [==============================] - 1038s 4s/step - loss: 0.3059 - val_loss: 0.2990
Epoch 97/100
254/254 [==============================] - 1077s 4s/step - loss: 0.3057 - val_loss: 0.2983
Epoch 98/100
254/254 [==============================] - 973s 4s/step - loss: 0.3058 - val_loss: 0.3006
Epoch 99/100
254/254 [==============================] - 894s 4s/step - loss: 0.3104 - val_loss: 0.3158
Epoch 100/100
254/254 [==============================] - 892s 4s/step - loss: 0.3156 - val_loss: 0.3054
```

```
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_co
lution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing
be directly callable after loading.
```

```
INFO:tensorflow:Assets written to: saved_model/my_model5_1\assets
```

```
INFO:tensorflow:Assets written to: saved_model/my_model5_1\assets
```

6.2 Testing

```
import numpy as np
import tensorflow as tf
import matplotlib. pyplot as plt
import os
from tensorflow. keras import layers
from tensorflow. keras. datasets import mnist
from tensorflow. keras. models import Model
from IPython.display import Image
import imageio
from PIL import Image
from skimage. transform import resize
```

```
import os
from os import listdir
import cv2
from PIL import Image
from skimage import color
from skimage import io
import pickle
def preprocess1(array):
    array = array. astype("float32") / 255.0
    array = np. reshape (array, (1, 400, 400,1))
    return array
new_model = tf. keras. models.load_model('my_model')
autoencoder = new_model
img =Image.open('input.tiff'). convert('L')
newsize = (400, 400)
img = img. resize(newsize)
np_img1 = np. array(img)
print(np_img1.shape)
print(np_img1.shape)
np_img1=preprocess1(np_img1)
print(np_img1.shape)
predictions2 = autoencoder. predict(np_img1)
n = 1
f = plt. figure (figsize=(400, 400))
y=0
for i in predictions2: # Debug, plot figure
    y=y+1
    f.add_subplot (10, 10, y)
    plt. gray ()
    plt. imshow(i)
plt. show(block=True)
for i in predictions2:
    im =i
    i = np. reshape (i, (400, 400))
    imageio. imwrite ('outfile5.tif', im)
    plt. imsave ("outfile5.tiff", i,cmap='gray'
    print (im. shape)
```

7. OUTPUT OF MODEL

- Original PSRN value of noisy image: **22.21**
- Final PSNR value of denoised image: **28.79**

In [2]:
```
from skimage import img_as_float
from skimage.metrics import peak_signal_noise_ratio
from matplotlib import pyplot as plt
from skimage import io
from scipy import ndimage as nd
import cv2
```

In [9]:
```
noisy_img = img_as_float(cv2.imread("Mnoiseimg.tiff",0))
#Need to convert to float as we will be doing math on the array
#Also, most skimage functions need float numbers
ref_img = img_as_float(cv2.imread("Mrealimg.tiff",0))
predicted=img_as_float(cv2.imread("outfile5.tiff",0))
```

In [6]:
```
#old model
noise_psnr = peak_signal_noise_ratio(ref_img, noisy_img)
gaussian_cleaned_psnr = peak_signal_noise_ratio(ref_img, predicted)
print("PSNR of input noisy image = ", noise_psnr)
print("PSNR of cleaned image = ", gaussian_cleaned_psnr)
```

```
PSNR of input noisy image =  22.219345330569695
PSNR of cleaned image =  27.14978261896217
```

In [8]:
```
#new model
noise_psnr = peak_signal_noise_ratio(ref_img, noisy_img)
gaussian_cleaned_psnr = peak_signal_noise_ratio(ref_img, predicted)
print("PSNR of input noisy image = ", noise_psnr)
print("PSNR of cleaned image = ", gaussian_cleaned_psnr)
```

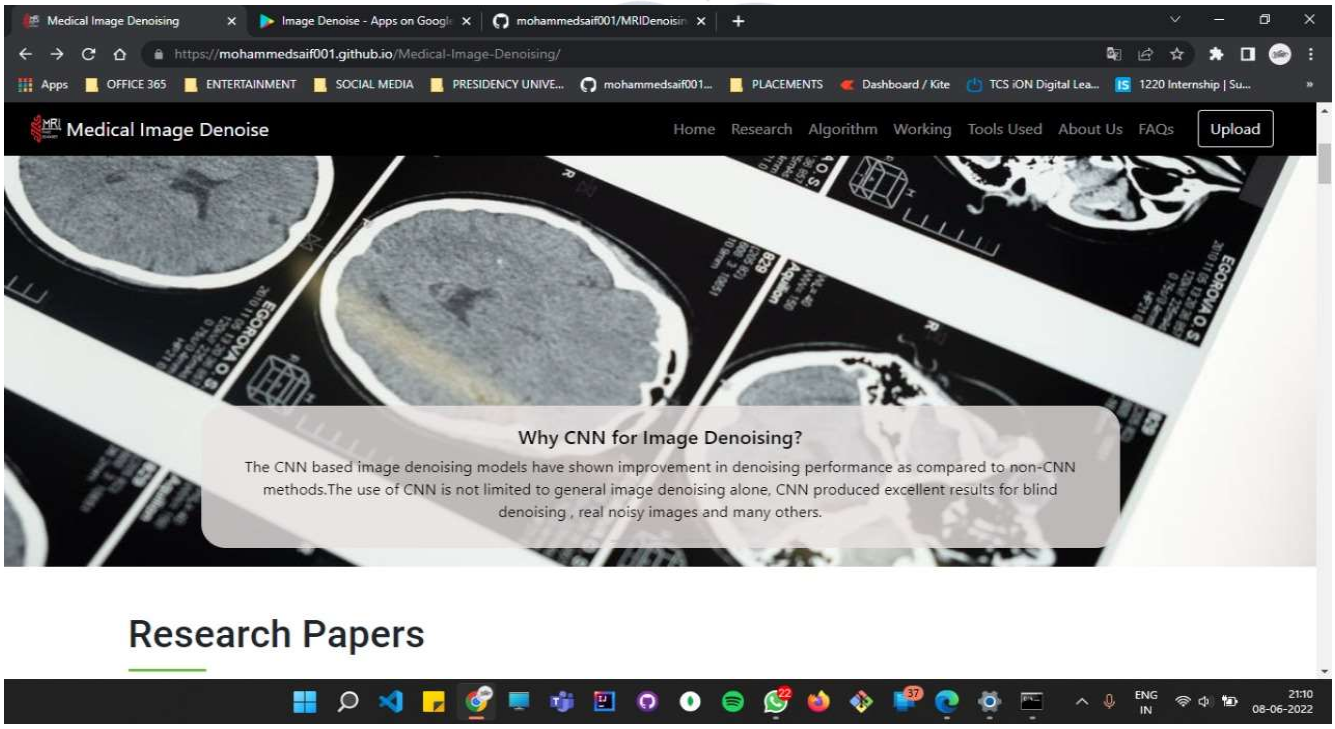
```
PSNR of input noisy image =  22.219345330569695
PSNR of cleaned image =  28.7935672397891
```

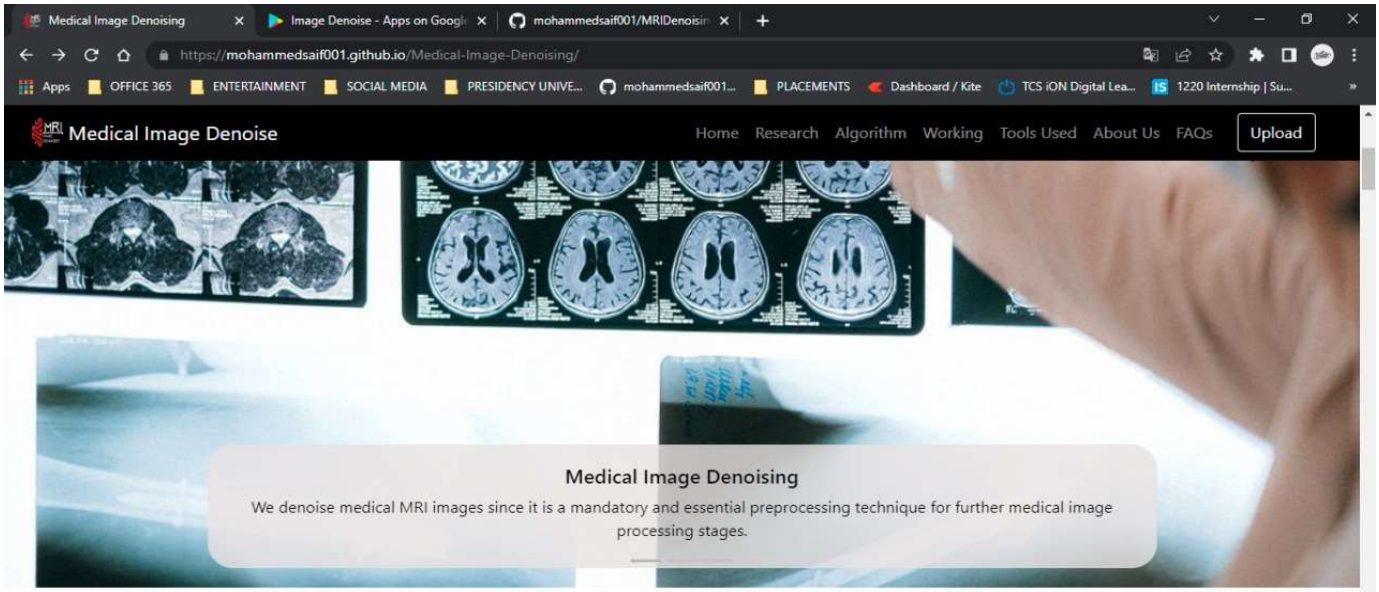
In [ ]:

8. APPLICATIONS IMPLEMENTED

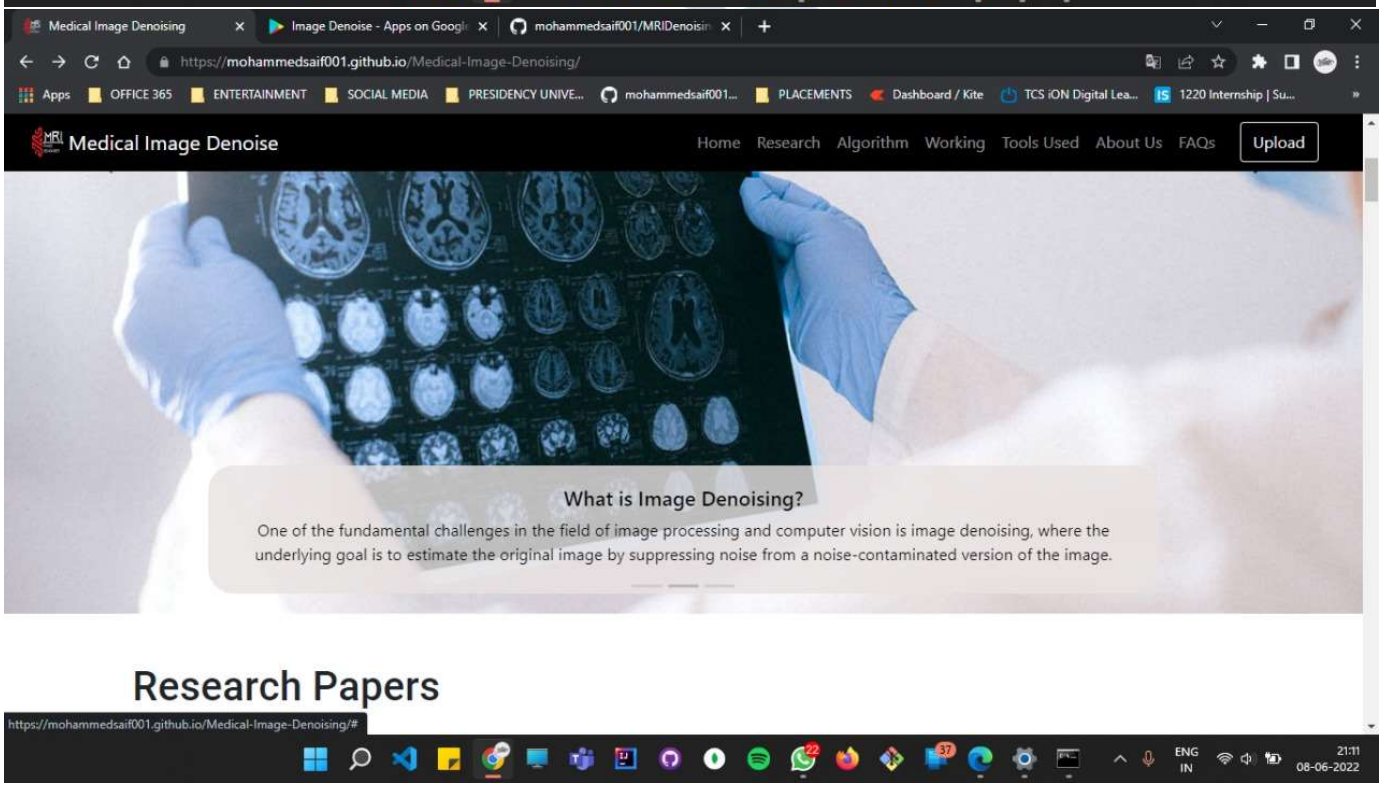
8.1 Image denoising Website

- **Link:** Medical Image Denoising (mohammedsaif001.github.io)
- **QR Code:**

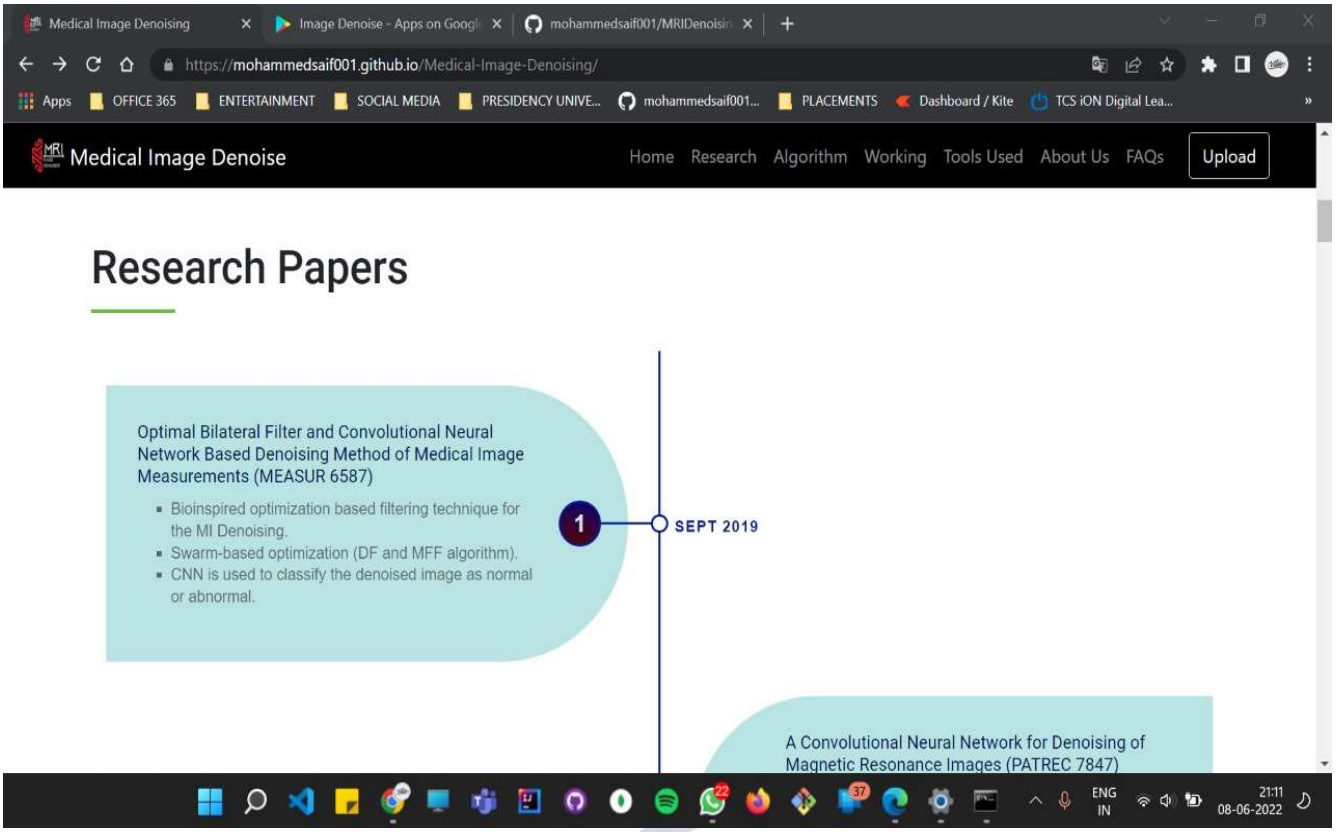


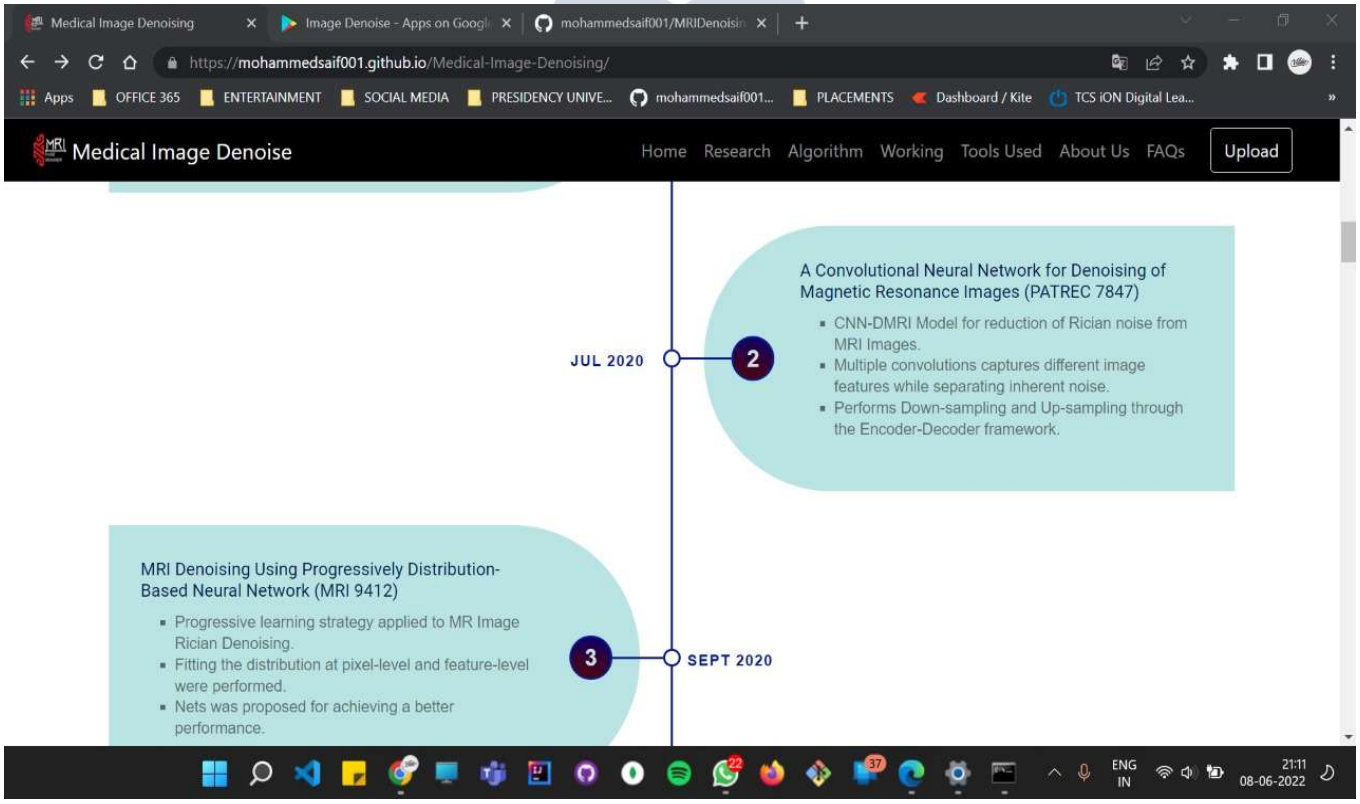


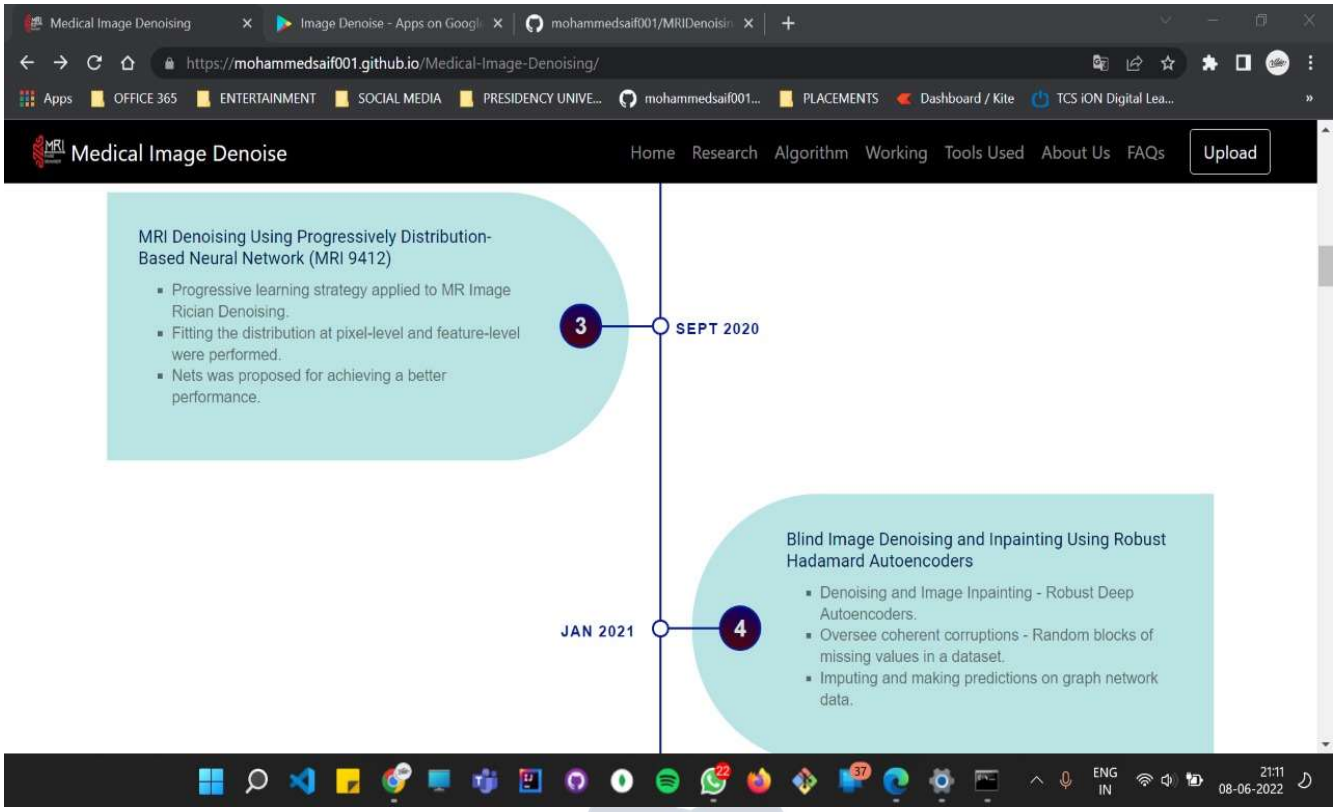
Research Papers

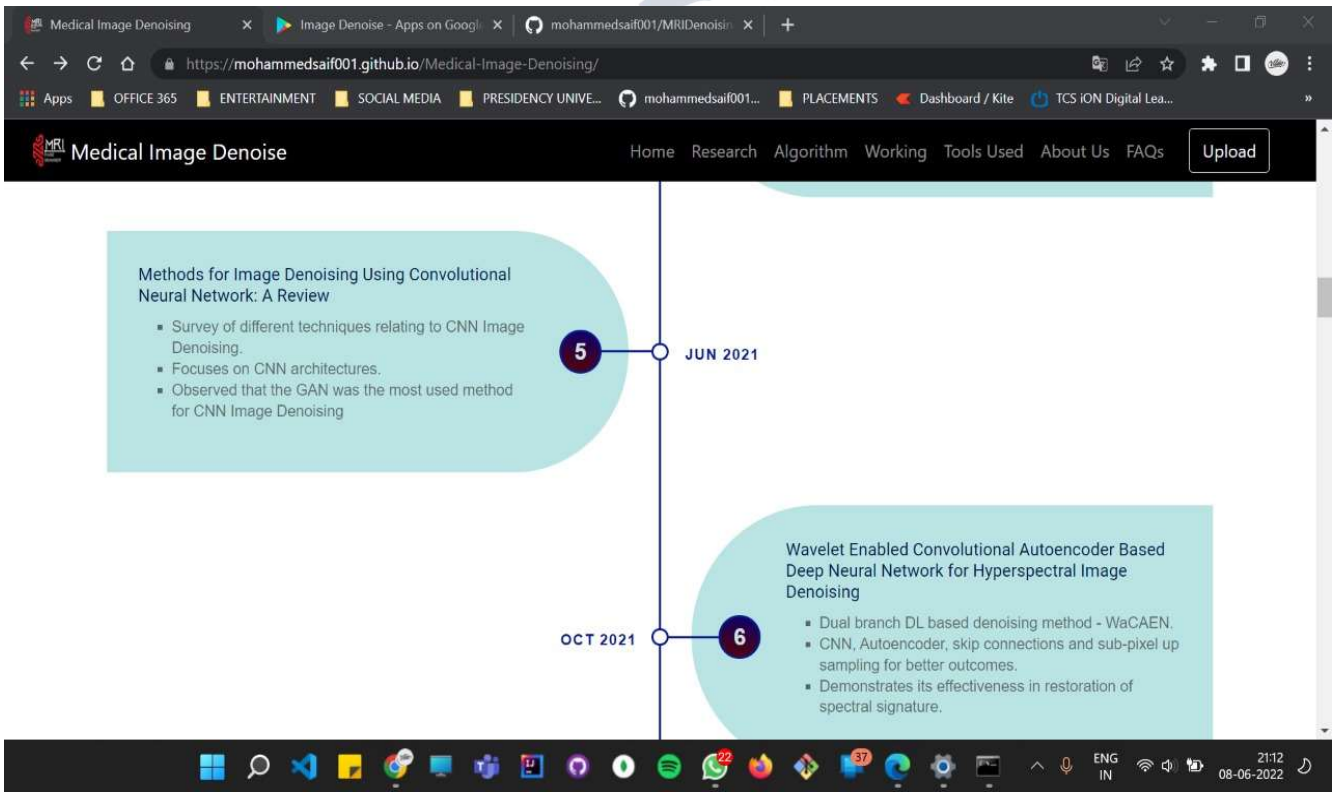


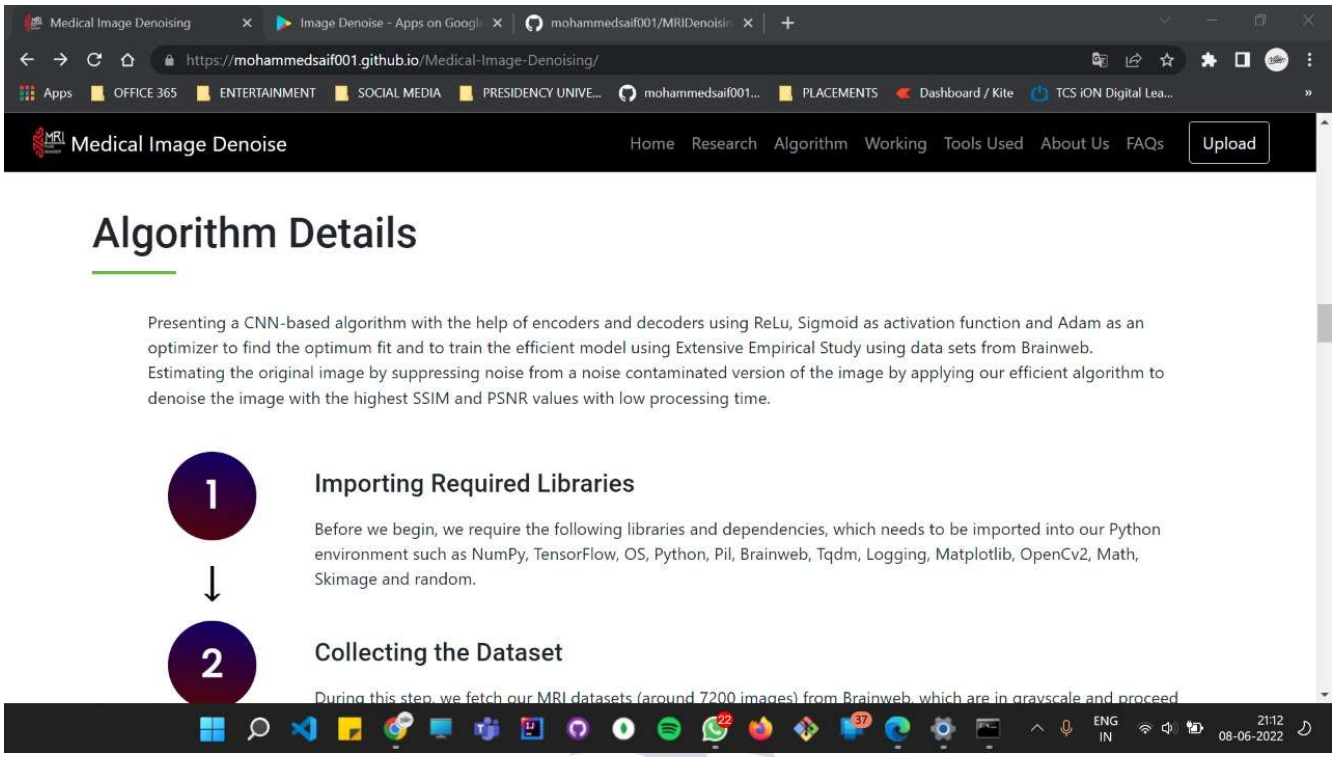
Research Papers

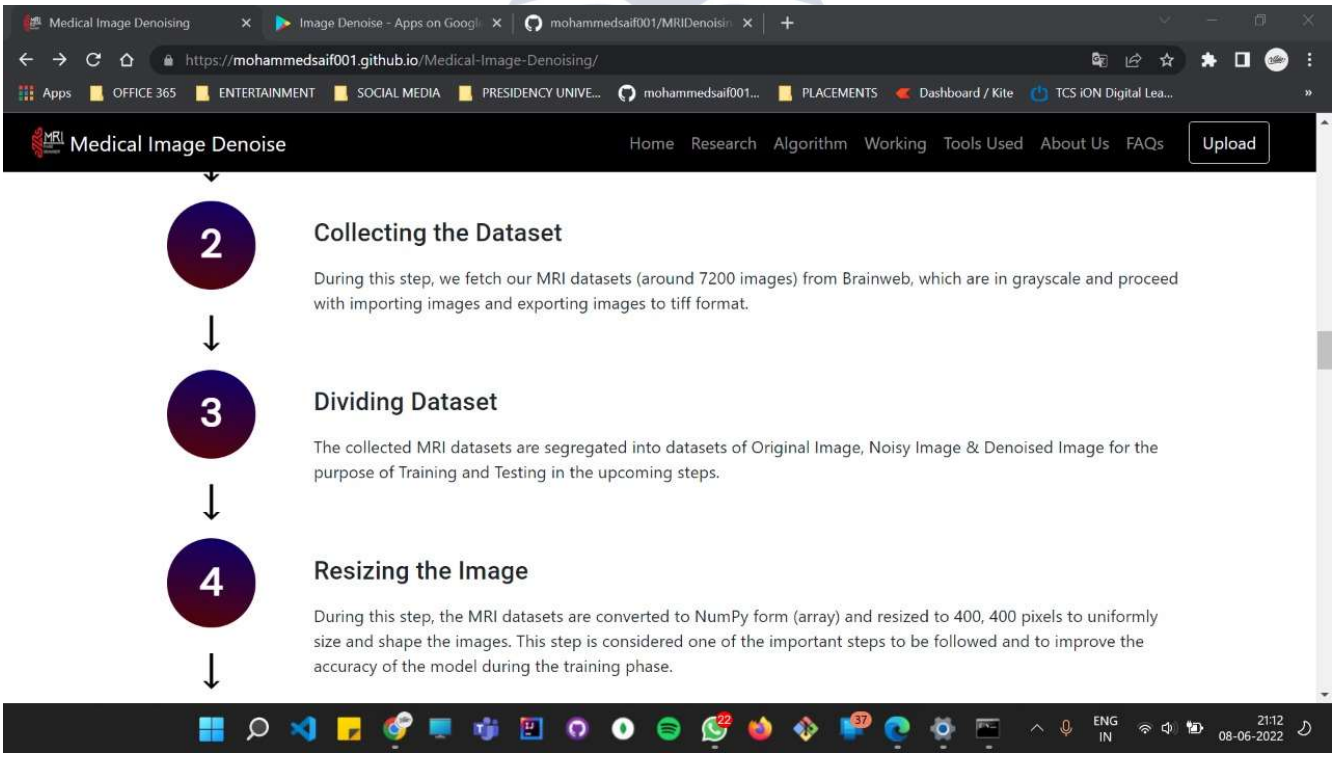

Medical Image Denoising
Image Denoise - Apps on Google
mohammedsaif001/MRIDenoising
https://mohammedsaif001.github.io/Medical-Image-Denoising/
Apps
OFFICE 365
ENTERTAINMENT
SOCIAL MEDIA
PRESIDENCY UNIVE...
mohammedsaif001...
PLACEMENTS
Dashboard / Kite
TCS iON Digital Lea...
Medical Image Denoise
Home
Research
Algorithm
Working
Tools Used
About Us
FAQs
Upload
Research Papers
Optimal Bilateral Filter and Convolutional Neural Network Based Denoising Method of Medical Image Measurements (MEASUR 6587)
Bioinspired optimization based filtering technique for the MI Denoising.
Swarm-based optimization (DF and MFF algorithm).
CNN is used to classify the denoised image as normal or abnormal.
1
SEPT 2019
A Convolutional Neural Network for Denoising of Magnetic Resonance Images (PATREC 7847)
ENG
IN
21:11
08-06-2022

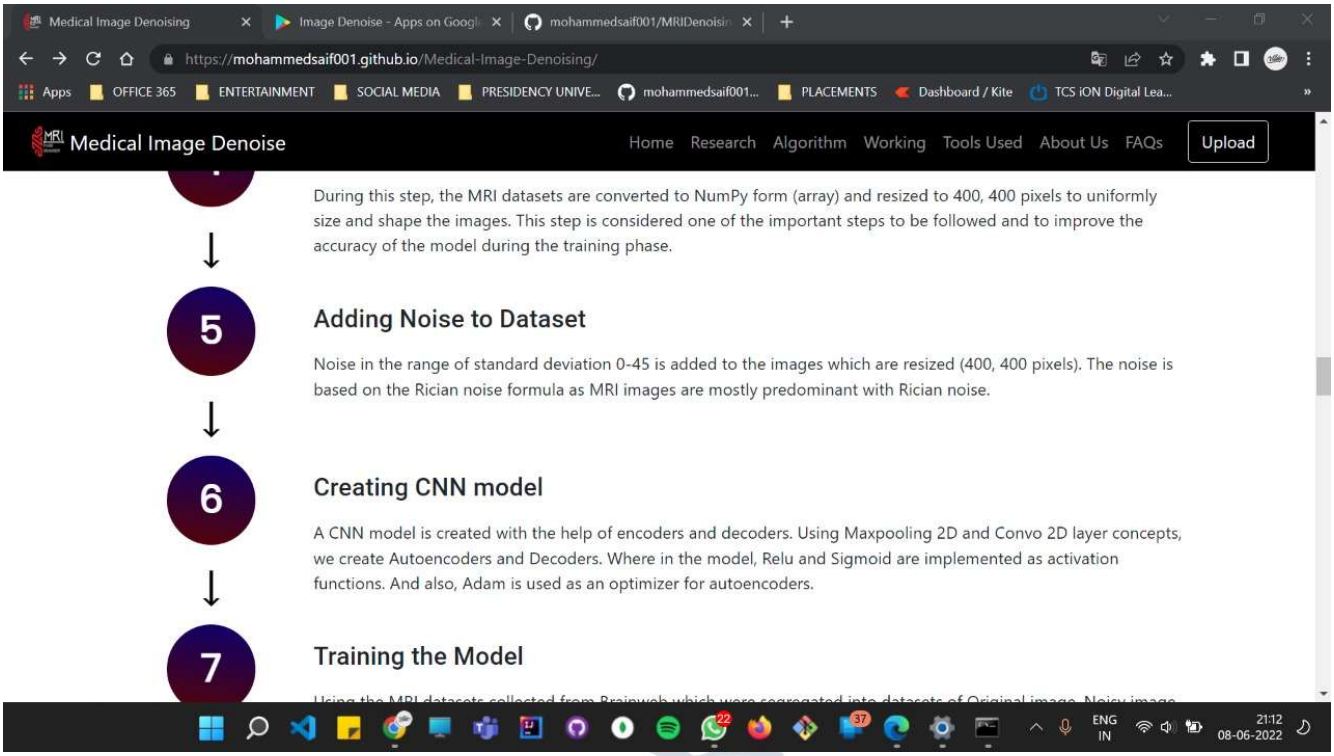

Medical Image Denoising
Image Denoise - Apps on Google
mohammedsaif001/MRIDenoising
https://mohammedsaif001.github.io/Medical-Image-Denoising/
Apps
OFFICE 365
ENTERTAINMENT
SOCIAL MEDIA
PRESIDENCY UNIVE...
mohammedsaif001...
PLACEMENTS
Dashboard / Kite
TCS iON Digital Lea...
Medical Image Denoise
Home
Research
Algorithm
Working
Tools Used
About Us
FAQs
Upload
JUL 2020
2
A Convolutional Neural Network for Denoising of Magnetic Resonance Images (PATREC 7847)
CNN-DMRI Model for reduction of Rician noise from MRI Images.
Multiple convolutions captures different image features while separating inherent noise.
Performs Down-sampling and Up-sampling through the Encoder-Decoder framework.
MRI Denoising Using Progressively Distribution-Based Neural Network (MRI 9412)
Progressive learning strategy applied to MR Image Rician Denoising.
Fitting the distribution at pixel-level and feature-level were performed.
Nets was proposed for achieving a better performance.
3
SEPT 2020
ENG
IN
21:11
08-06-2022

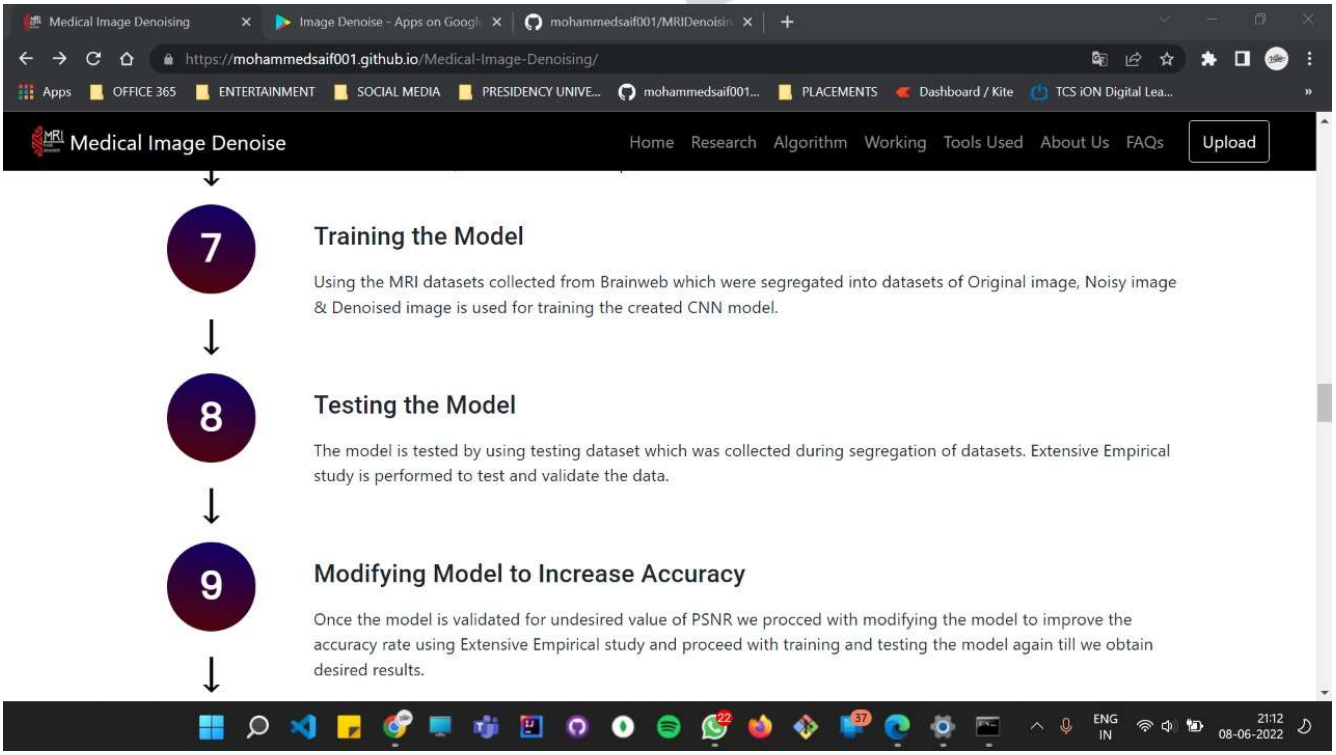

Medical Image Denoising
Image Denoise - Apps on Google
mohammedsaif001/MRIDenoising
https://mohammedsaif001.github.io/Medical-Image-Denoising/
Apps
OFFICE 365
ENTERTAINMENT
SOCIAL MEDIA
PRESIDENCY UNIVE...
mohammedsaif001...
PLACEMENTS
Dashboard / Kite
TCS iON Digital Lea...
Medical Image Denoise
Home
Research
Algorithm
Working
Tools Used
About Us
FAQs
Upload
MRI Denoising Using Progressively Distribution-Based Neural Network (MRI 9412)
Progressive learning strategy applied to MR Image Rician Denoising.
Fitting the distribution at pixel-level and feature-level were performed.
Nets was proposed for achieving a better performance.
3
SEPT 2020
JAN 2021
4
Blind Image Denoising and Inpainting Using Robust Hadamard Autoencoders
Denoising and Image Inpainting - Robust Deep Autoencoders.
Oversee coherent corruptions - Random blocks of missing values in a dataset.
Imputing and making predictions on graph network data.
ENG
IN
21:11
08-06-2022

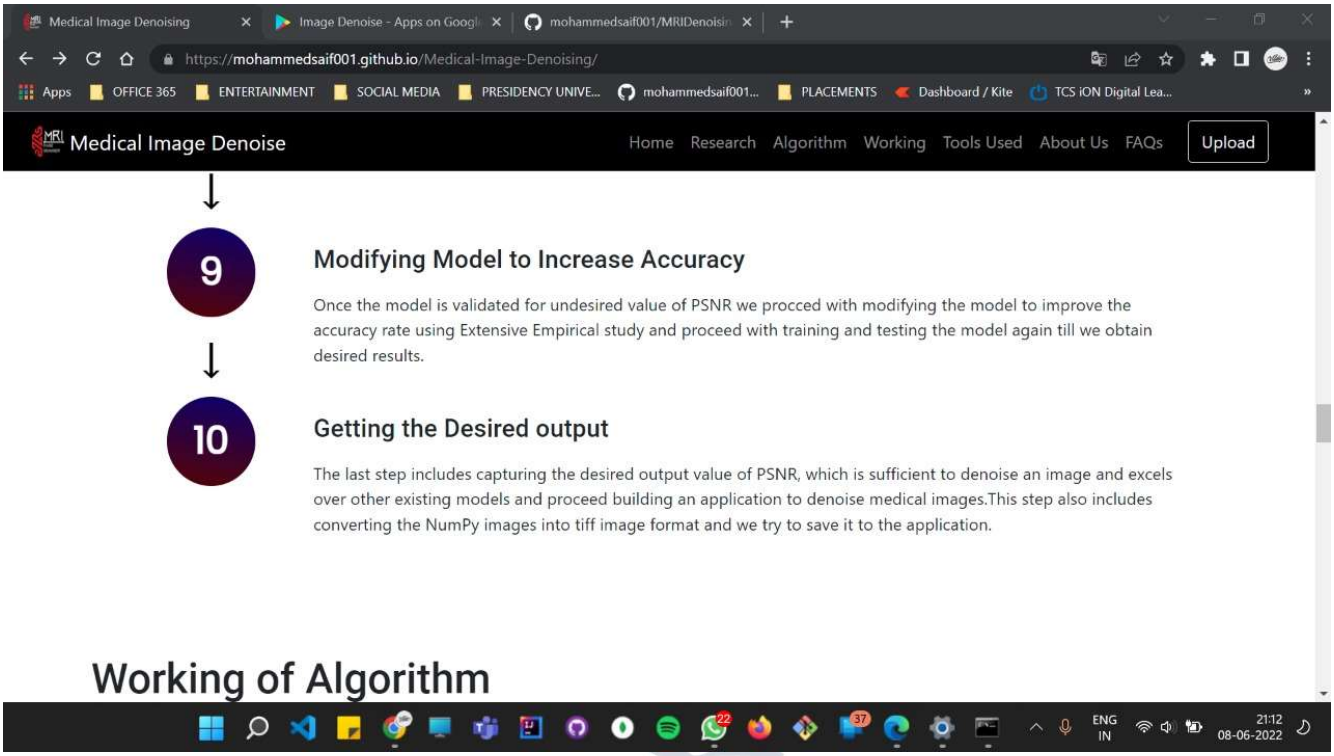

Medical Image Denoising
Image Denoise - Apps on Google
mohammedsaif001/MRIDenoising
https://mohammedsaif001.github.io/Medical-Image-Denoising/
Apps
OFFICE 365
ENTERTAINMENT
SOCIAL MEDIA
PRESIDENCY UNIVE...
mohammedsaif001...
PLACEMENTS
Dashboard / Kite
TCS iON Digital Lea...
Medical Image Denoise
Home
Research
Algorithm
Working
Tools Used
About Us
FAQs
Upload
Methods for Image Denoising Using Convolutional Neural Network: A Review
Survey of different techniques relating to CNN Image Denoising.
Focuses on CNN architectures.
Observed that the GAN was the most used method for CNN Image Denoising
5
JUN 2021
OCT 2021
6
Wavelet Enabled Convolutional Autoencoder Based Deep Neural Network for Hyperspectral Image Denoising
Dual branch DL based denoising method - WaCAEN.
CNN, Autoencoder, skip connections and sub-pixel up sampling for better outcomes.
Demonstrates its effectiveness in restoration of spectral signature.
ENG
IN
21:12
08-06-2022

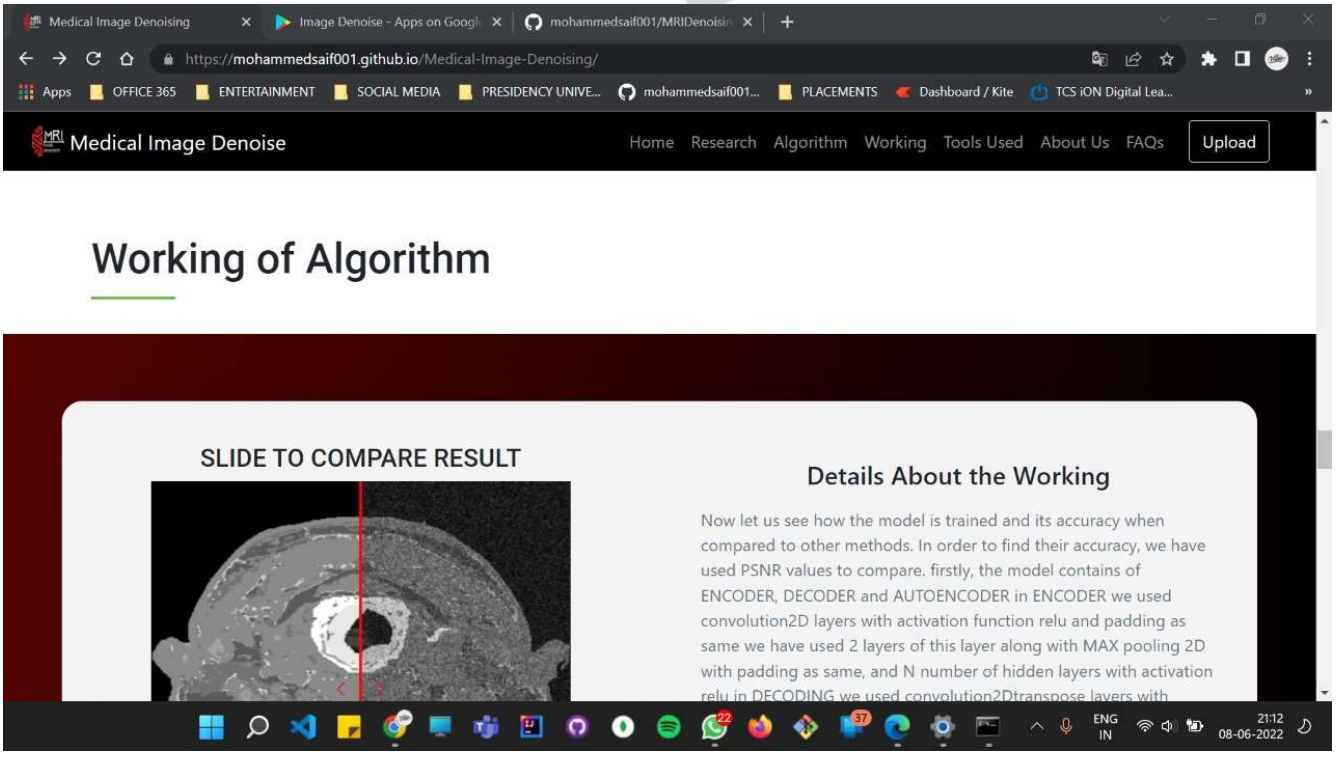

Medical Image Denoising
Image Denoise - Apps on Googl
mohammedsaif001/MRIDenoisin
https://mohammedsaif001.github.io/Medical-Image-Denoising/
Apps
OFFICE 365
ENTERTAINMENT
SOCIAL MEDIA
PRESIDENCY UNIVE...
mohammedsaif001...
PLACEMENTS
Dashboard / Kite
TCS iON Digital Lea...
Medical Image Denoise
Home
Research
Algorithm
Working
Tools Used
About Us
FAQs
Upload
Algorithm Details
Presenting a CNN-based algorithm with the help of encoders and decoders using ReLu, Sigmoid as activation function and Adam as an optimizer to find the optimum fit and to train the efficient model using Extensive Empirical Study using data sets from Brainweb. Estimating the original image by suppressing noise from a noise contaminated version of the image by applying our efficient algorithm to denoise the image with the highest SSIM and PSNR values with low processing time.
1
Importing Required Libraries
Before we begin, we require the following libraries and dependencies, which needs to be imported into our Python environment such as NumPy, TensorFlow, OS, Python, Pil, Brainweb, Tqdm, Logging, Matplotlib, OpenCv2, Math, Skimage and random.
2
Collecting the Dataset
ENG
IN
21:12
08-06-2022

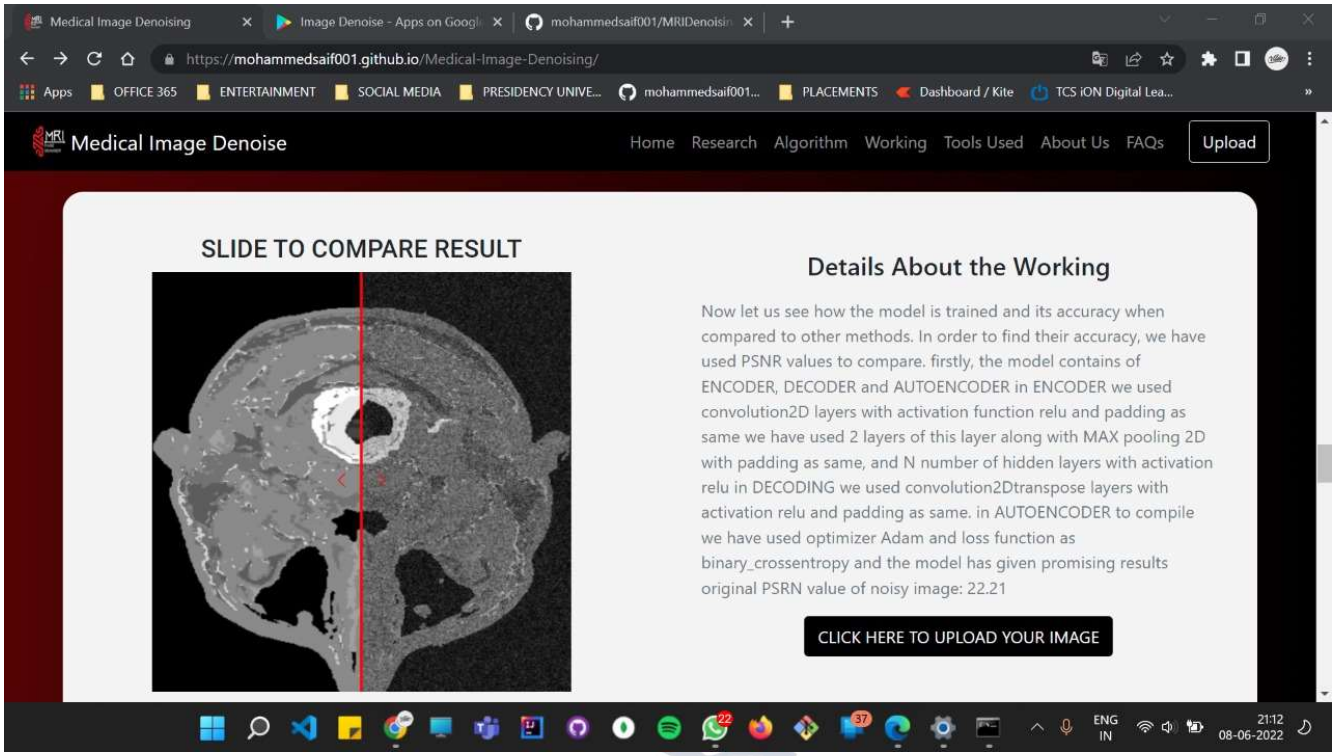

Medical Image Denoising
Image Denoise - Apps on Googl
mohammedsaif001/MRIDenoisin
https://mohammedsaif001.github.io/Medical-Image-Denoising/
Apps
OFFICE 365
ENTERTAINMENT
SOCIAL MEDIA
PRESIDENCY UNIVE...
mohammedsaif001...
PLACEMENTS
Dashboard / Kite
TCS iON Digital Lea...
Medical Image Denoise
Home
Research
Algorithm
Working
Tools Used
About Us
FAQs
Upload
2
Collecting the Dataset
During this step, we fetch our MRI datasets (around 7200 images) from Brainweb, which are in grayscale and proceed with importing images and exporting images to tiff format.
3
Dividing Dataset
The collected MRI datasets are segregated into datasets of Original Image, Noisy Image & Denoised Image for the purpose of Training and Testing in the upcoming steps.
4
Resizing the Image
During this step, the MRI datasets are converted to NumPy form (array) and resized to 400, 400 pixels to uniformly size and shape the images. This step is considered one of the important steps to be followed and to improve the accuracy of the model during the training phase.
ENG
IN
21:12
08-06-2022

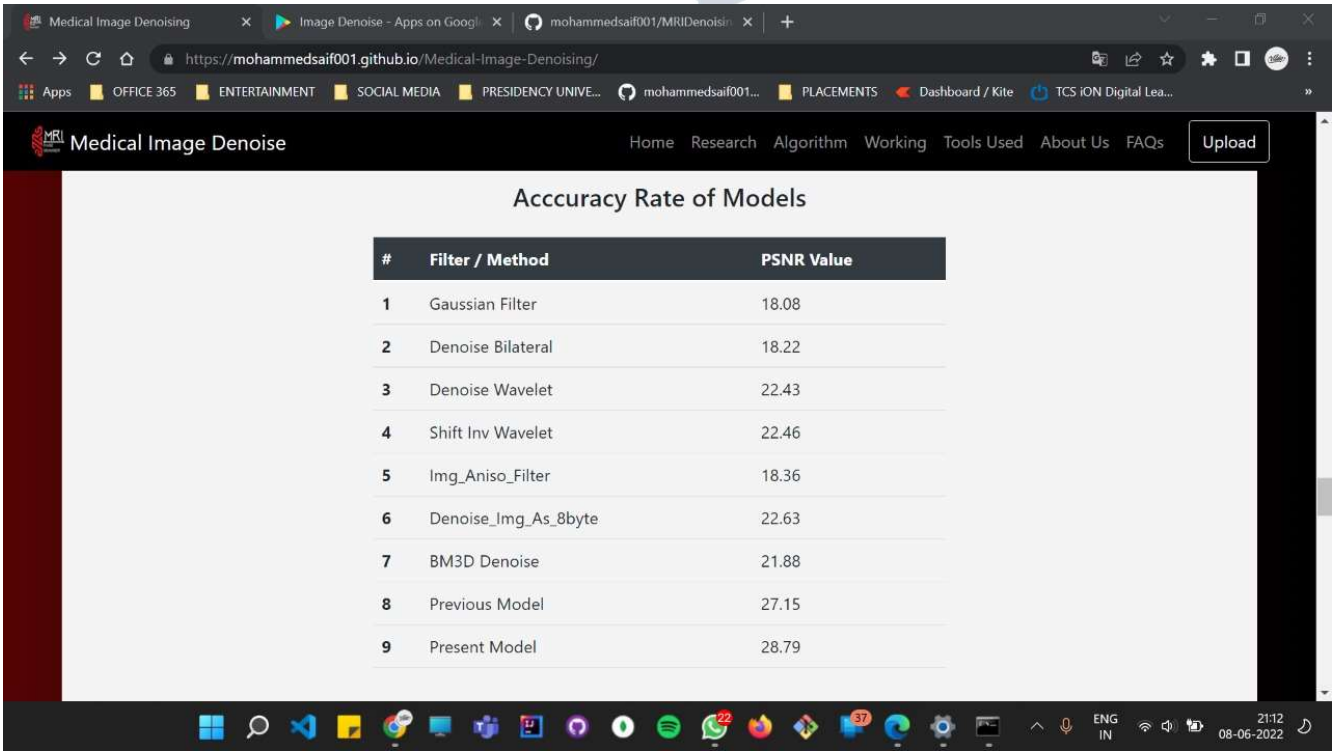

Medical Image Denoising
Image Denoise - Apps on Googl
mohammedsaif001/MRIDenoisin
https://mohammedsaif001.github.io/Medical-Image-Denoising/
Apps
OFFICE 365
ENTERTAINMENT
SOCIAL MEDIA
PRESIDENCY UNIVE...
mohammedsaif001...
PLACEMENTS
Dashboard / Kite
TCS iON Digital Lea...
Medical Image Denoise
Home
Research
Algorithm
Working
Tools Used
About Us
FAQs
Upload
During this step, the MRI datasets are converted to NumPy form (array) and resized to 400, 400 pixels to uniformly size and shape the images. This step is considered one of the important steps to be followed and to improve the accuracy of the model during the training phase.
5
Adding Noise to Dataset
Noise in the range of standard deviation 0-45 is added to the images which are resized (400, 400 pixels). The noise is based on the Rician noise formula as MRI images are mostly predominant with Rician noise.
6
Creating CNN model
A CNN model is created with the help of encoders and decoders. Using Maxpooling 2D and Convo 2D layer concepts, we create Autoencoders and Decoders. Where in the model, Relu and Sigmoid are implemented as activation functions. And also, Adam is used as an optimizer for autoencoders.
7
Training the Model
ENG
IN
21:12
08-06-2022

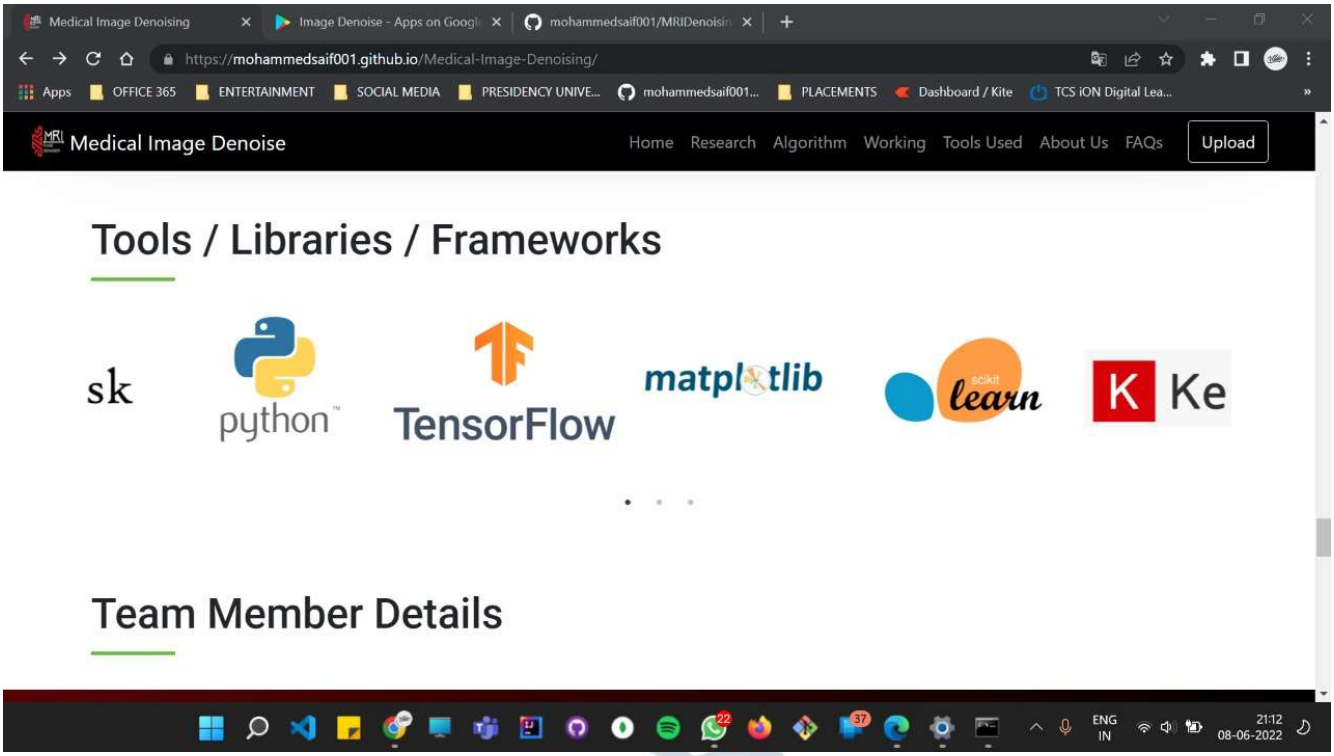

Medical Image Denoising
Image Denoise - Apps on Googl
mohammedsaif001/MRIDenoisin
https://mohammedsaif001.github.io/Medical-Image-Denoising/
Apps
OFFICE 365
ENTERTAINMENT
SOCIAL MEDIA
PRESIDENCY UNIVE...
mohammedsaif001...
PLACEMENTS
Dashboard / Kite
TCS iON Digital Lea...
Medical Image Denoise
Home
Research
Algorithm
Working
Tools Used
About Us
FAQs
Upload
7
Training the Model
Using the MRI datasets collected from Brainweb which were segregated into datasets of Original image, Noisy image & Denoised image is used for training the created CNN model.
8
Testing the Model
The model is tested by using testing dataset which was collected during segregation of datasets. Extensive Empirical study is performed to test and validate the data.
9
Modifying Model to Increase Accuracy
Once the model is validated for undesired value of PSNR we procced with modifying the model to improve the accuracy rate using Extensive Empirical study and proceed with training and testing the model again till we obtain desired results.
ENG
IN
21:12
08-06-2022

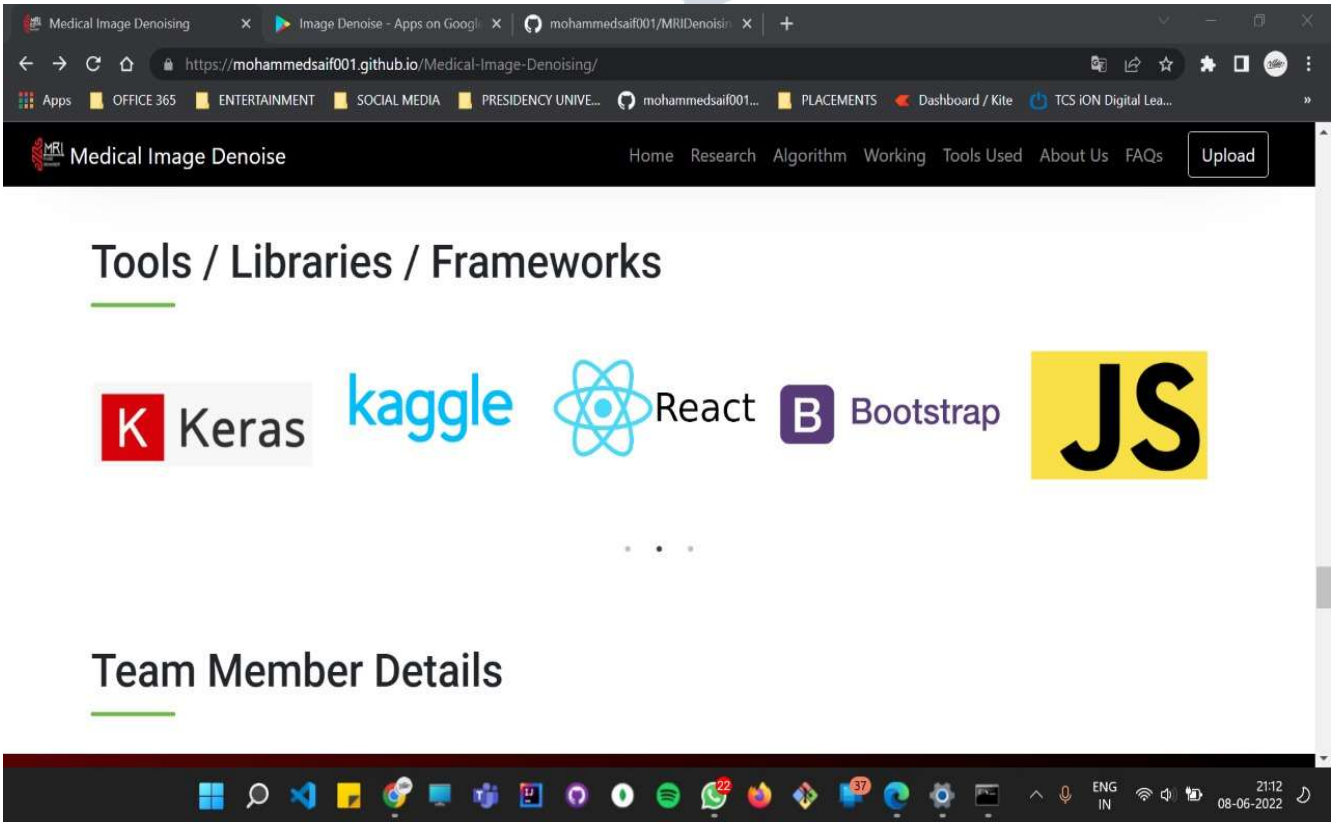

Medical Image Denoising
Image Denoise - Apps on Googl
mohammedsaif001/MRIDenoisin
https://mohammedsaif001.github.io/Medical-Image-Denoising/
Apps
OFFICE 365
ENTERTAINMENT
SOCIAL MEDIA
PRESIDENCY UNIVE...
mohammedsaif001...
PLACEMENTS
Dashboard / Kite
TCS iON Digital Lea...
Medical Image Denoise
Home
Research
Algorithm
Working
Tools Used
About Us
FAQs
Upload
9
Modifying Model to Increase Accuracy
Once the model is validated for undesired value of PSNR we procced with modifying the model to improve the accuracy rate using Extensive Empirical study and proceed with training and testing the model again till we obtain desired results.
10
Getting the Desired output
The last step includes capturing the desired output value of PSNR, which is sufficient to denoise an image and excels over other existing models and proceed building an application to denoise medical images.This step also includes converting the NumPy images into tiff image format and we try to save it to the application.
Working of Algorithm
ENG
IN
21:12
08-06-2022

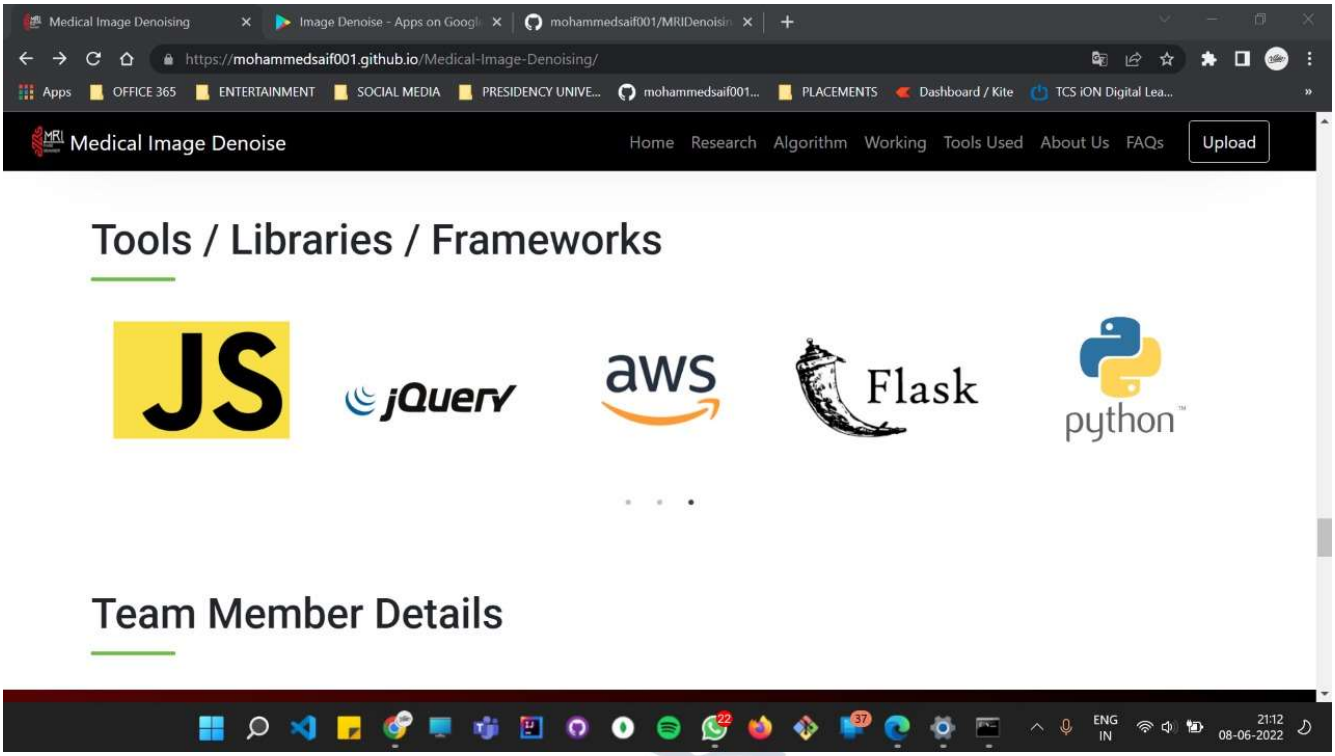

Medical Image Denoising
Image Denoise - Apps on Googl
mohammedsaif001/MRIDenoisin
https://mohammedsaif001.github.io/Medical-Image-Denoising/
Apps
OFFICE 365
ENTERTAINMENT
SOCIAL MEDIA
PRESIDENCY UNIVE...
mohammedsaif001...
PLACEMENTS
Dashboard / Kite
TCS iON Digital Lea...
Medical Image Denoise
Home
Research
Algorithm
Working
Tools Used
About Us
FAQs
Upload
Working of Algorithm
SLIDE TO COMPARE RESULT
Details About the Working
Now let us see how the model is trained and its accuracy when compared to other methods. In order to find their accuracy, we have used PSNR values to compare. firstly, the model contains of ENCODER, DECODER and AUTOENCODER in ENCODER we used convolution2D layers with activation function relu and padding as same we have used 2 layers of this layer along with MAX pooling 2D with padding as same, and N number of hidden layers with activation
ENG
IN
21:12
08-06-2022

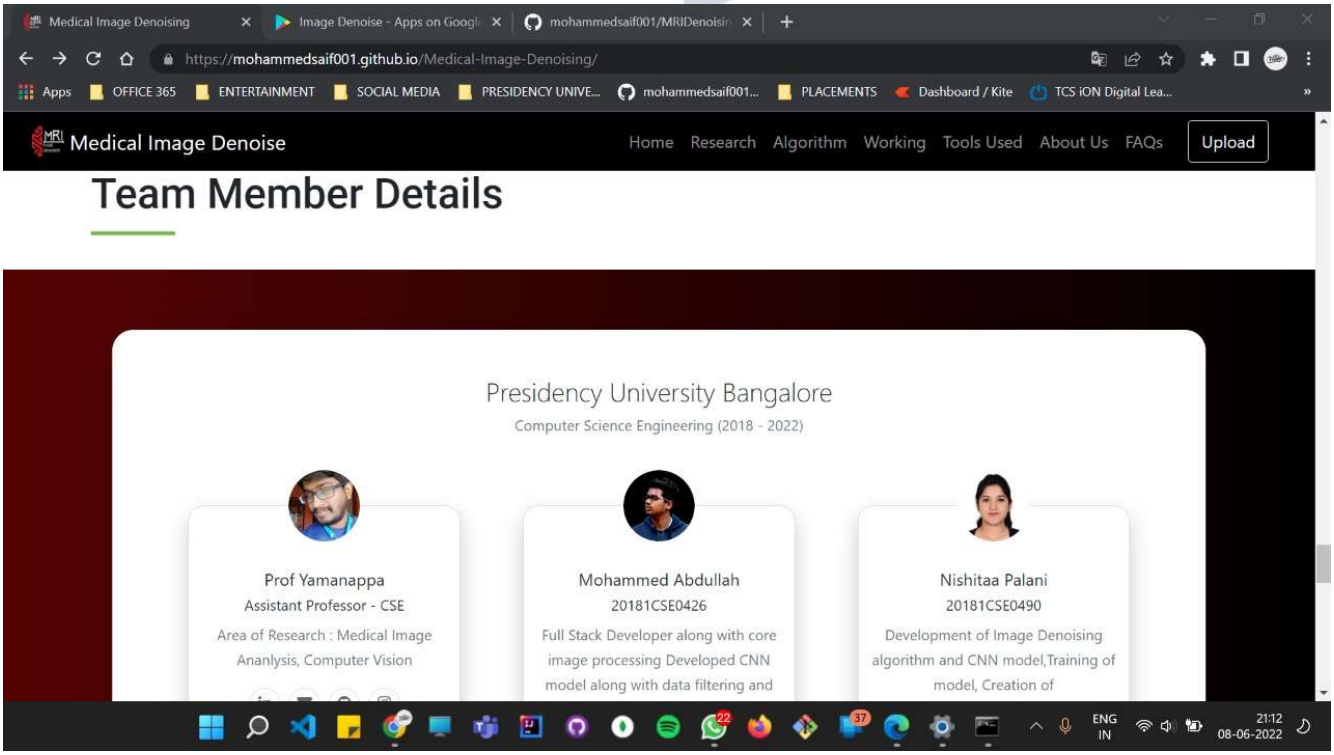

Medical Image Denoising
Image Denoise - Apps on Google
mohammedsaif001/MRIDenoising
https://mohammedsaif001.github.io/Medical-Image-Denoising/
Apps
OFFICE 365
ENTERTAINMENT
SOCIAL MEDIA
PRESIDENCY UNIVE...
mohammedsaif001...
PLACEMENTS
Dashboard / Kite
TCS iON Digital Lea...
Medical Image Denoise
Home
Research
Algorithm
Working
Tools Used
About Us
FAQs
Upload
SLIDE TO COMPARE RESULT
Details About the Working
Now let us see how the model is trained and its accuracy when compared to other methods. In order to find their accuracy, we have used PSNR values to compare. firstly, the model contains of ENCODER, DECODER and AUTOENCODER in ENCODER we used convolution2D layers with activation function relu and padding as same we have used 2 layers of this layer along with MAX pooling 2D with padding as same, and N number of hidden layers with activation relu in DECODING we used convolution2Dtranspose layers with activation relu and padding as same. in AUTOENCODER to compile we have used optimizer Adam and loss function as binary_crossentropy and the model has given promising results original PSRN value of noisy image: 22.21
CLICK HERE TO UPLOAD YOUR IMAGE
ENG
IN
21:12
08-06-2022

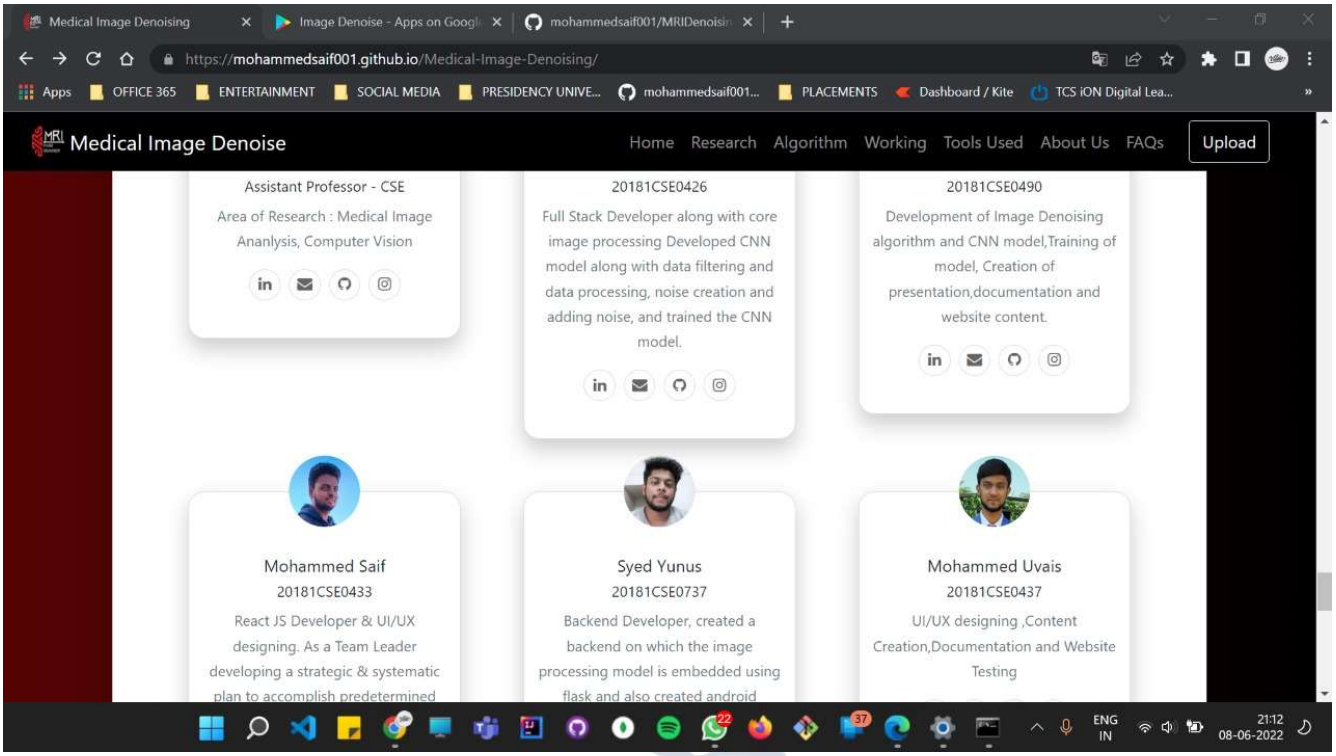

Medical Image Denoising
Image Denoise - Apps on Google
mohammedsaif001/MRIDenoising
https://mohammedsaif001.github.io/Medical-Image-Denoising/
Apps
OFFICE 365
ENTERTAINMENT
SOCIAL MEDIA
PRESIDENCY UNIVE...
mohammedsaif001...
PLACEMENTS
Dashboard / Kite
TCS iON Digital Lea...
Medical Image Denoise
Home
Research
Algorithm
Working
Tools Used
About Us
FAQs
Upload
Acccuracy Rate of Models
Filter / Method PSNR Value
1 Gaussian Filter 18.08
2 Denoise Bilateral 18.22
3 Denoise Wavelet 22.43
4 Shift Inv Wavelet 22.46
5 Img_Aniso_Filter 18.36
6 Denoise_Img_As_8byte 22.63
7 BM3D Denoise 21.88
8 Previous Model 27.15
9 Present Model 28.79
ENG
IN
21:12
08-06-2022

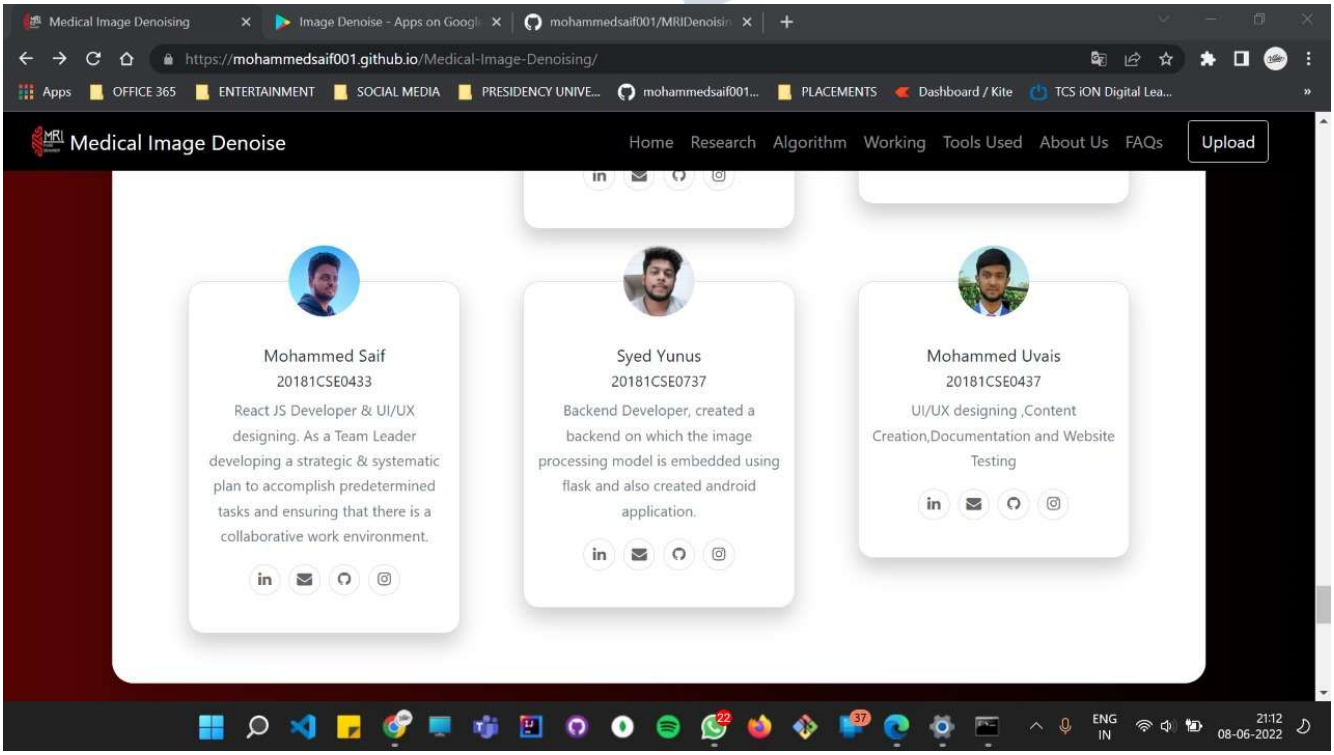

Medical Image Denoising
Image Denoise - Apps on Google
mohammedsaif001/MRIDenoising
https://mohammedsaif001.github.io/Medical-Image-Denoising/
Apps
OFFICE 365
ENTERTAINMENT
SOCIAL MEDIA
PRESIDENCY UNIVE...
mohammedsaif001...
PLACEMENTS
Dashboard / Kite
TCS iON Digital Lea...
Medical Image Denoise
Home
Research
Algorithm
Working
Tools Used
About Us
FAQs
Upload
Tools / Libraries / Frameworks
sk
python
TensorFlow
matplotlib
scikit learn
K Ke
Team Member Details
ENG
IN
21:12
08-06-2022

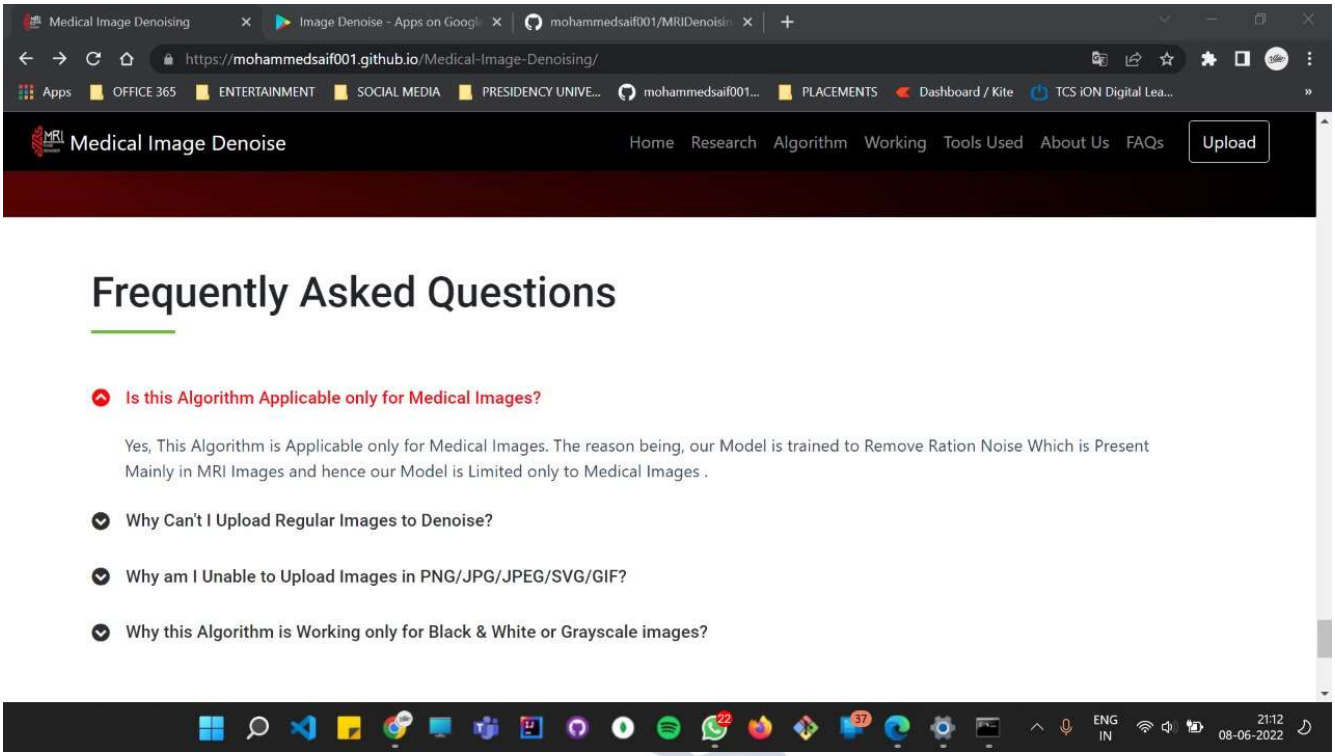

Medical Image Denoising
Image Denoise - Apps on Google
mohammedsaif001/MRIDenoising
https://mohammedsaif001.github.io/Medical-Image-Denoising/
Apps
OFFICE 365
ENTERTAINMENT
SOCIAL MEDIA
PRESIDENCY UNIVE...
mohammedsaif001...
PLACEMENTS
Dashboard / Kite
TCS iON Digital Lea...
Medical Image Denoise
Home
Research
Algorithm
Working
Tools Used
About Us
FAQs
Upload
Tools / Libraries / Frameworks
K Keras
kaggle
React
B Bootstrap
JS
Team Member Details
ENG
IN
21:12
08-06-2022

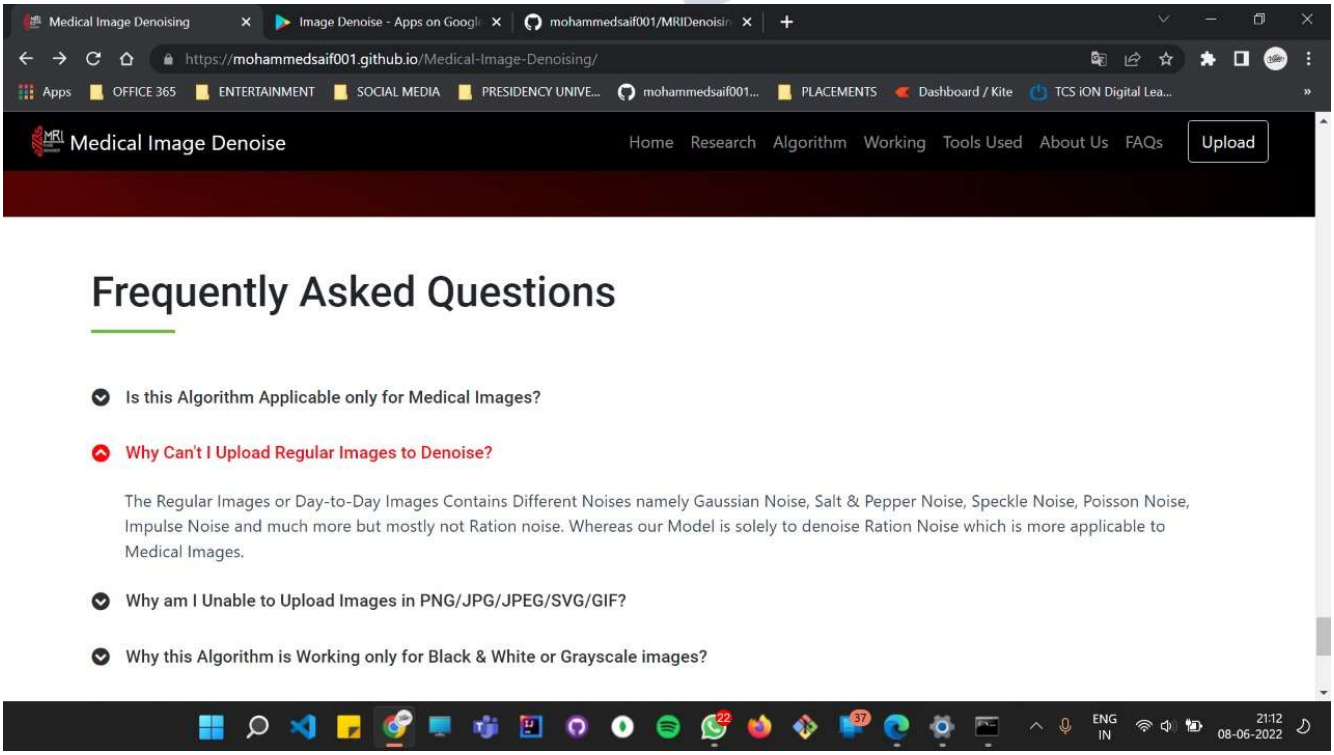

Medical Image Denoising
Image Denoise - Apps on Google
mohammedsaif001/MRIDenoising
https://mohammedsaif001.github.io/Medical-Image-Denoising/
Apps
OFFICE 365
ENTERTAINMENT
SOCIAL MEDIA
PRESIDENCY UNIVE...
mohammedsaif001...
PLACEMENTS
Dashboard / Kite
TCS iON Digital Lea...
Medical Image Denoise
Home
Research
Algorithm
Working
Tools Used
About Us
FAQs
Upload
Tools / Libraries / Frameworks
JS
jQuery
aws
Flask
python
Team Member Details
ENG
IN
21:12
08-06-2022

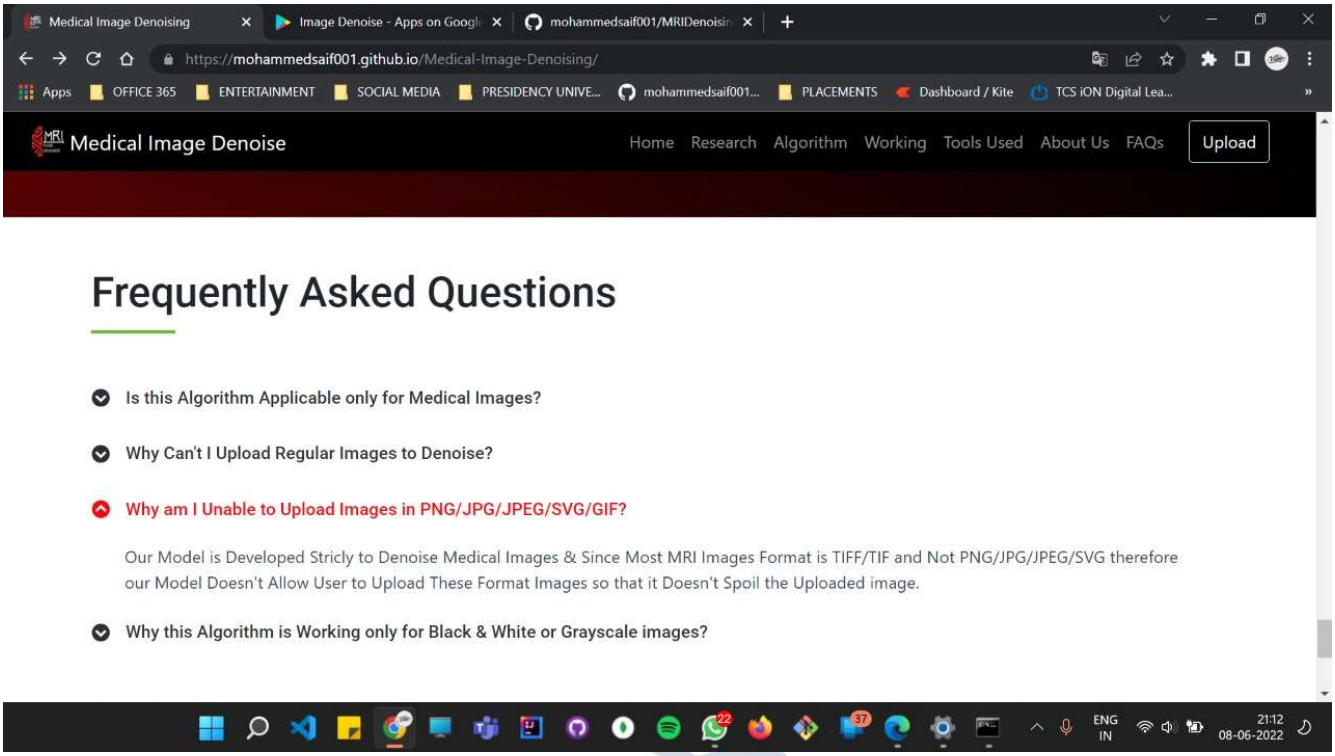

Medical Image Denoising
Image Denoise - Apps on Google
mohammedsaif001/MRIDenoising
https://mohammedsaif001.github.io/Medical-Image-Denoising/
Apps
OFFICE 365
ENTERTAINMENT
SOCIAL MEDIA
PRESIDENCY UNIVE...
mohammedsaif001...
PLACEMENTS
Dashboard / Kite
TCS iON Digital Lea...
Medical Image Denoise
Home
Research
Algorithm
Working
Tools Used
About Us
FAQs
Upload
Team Member Details
Presidency University Bangalore
Computer Science Engineering (2018 - 2022)
Prof Yamanappa
Assistant Professor - CSE
Area of Research : Medical Image Ananlysis, Computer Vision
Mohammed Abdullah
20181CSE0426
Full Stack Developer along with core image processing Developed CNN model along with data filtering and
Nishitaa Palani
20181CSE0490
Development of Image Denoising algorithm and CNN model,Training of model, Creation of
ENG
IN
21:12
08-06-2022

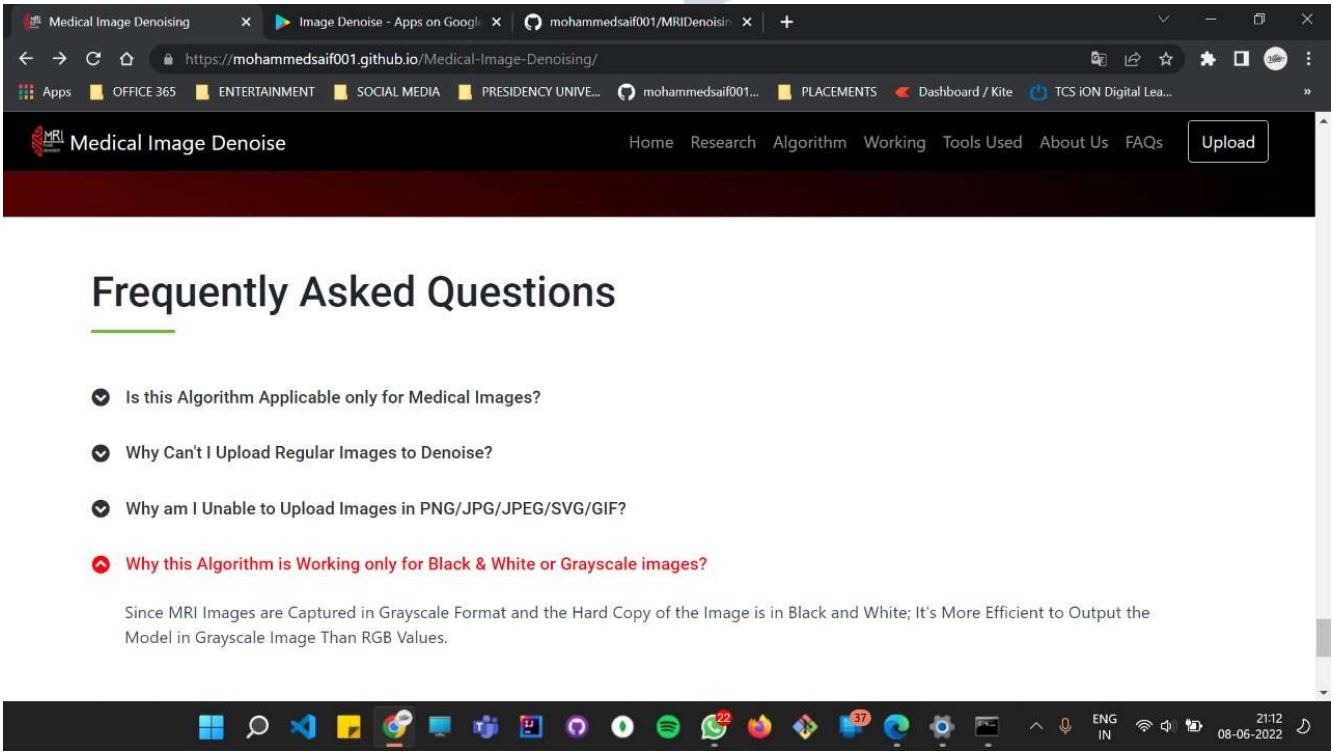

Medical Image Denoising
Image Denoise - Apps on Google
mohammedsaif001/MRIDenoising
https://mohammedsaif001.github.io/Medical-Image-Denoising/
Apps
OFFICE 365
ENTERTAINMENT
SOCIAL MEDIA
PRESIDENCY UNIVE...
mohammedsaif001...
PLACEMENTS
Dashboard / Kite
TCS iON Digital Lea...
Medical Image Denoise
Home
Research
Algorithm
Working
Tools Used
About Us
FAQs
Upload
Assistant Professor - CSE
Area of Research : Medical Image Ananlysis, Computer Vision
20181CSE0426
Full Stack Developer along with core image processing Developed CNN model along with data filtering and data processing, noise creation and adding noise, and trained the CNN model.
20181CSE0490
Development of Image Denoising algorithm and CNN model,Training of model, Creation of presentation,documentation and website content.
Mohammed Saif
20181CSE0433
React JS Developer & UI/UX designing. As a Team Leader developing a strategic & systematic plan to accomplish predetermined
Syed Yunus
20181CSE0737
Backend Developer, created a backend on which the image processing model is embedded using flask and also created android
Mohammed Uvais
20181CSE0437
UI/UX designing ,Content Creation,Documentation and Website Testing
ENG
IN
21:12
08-06-2022

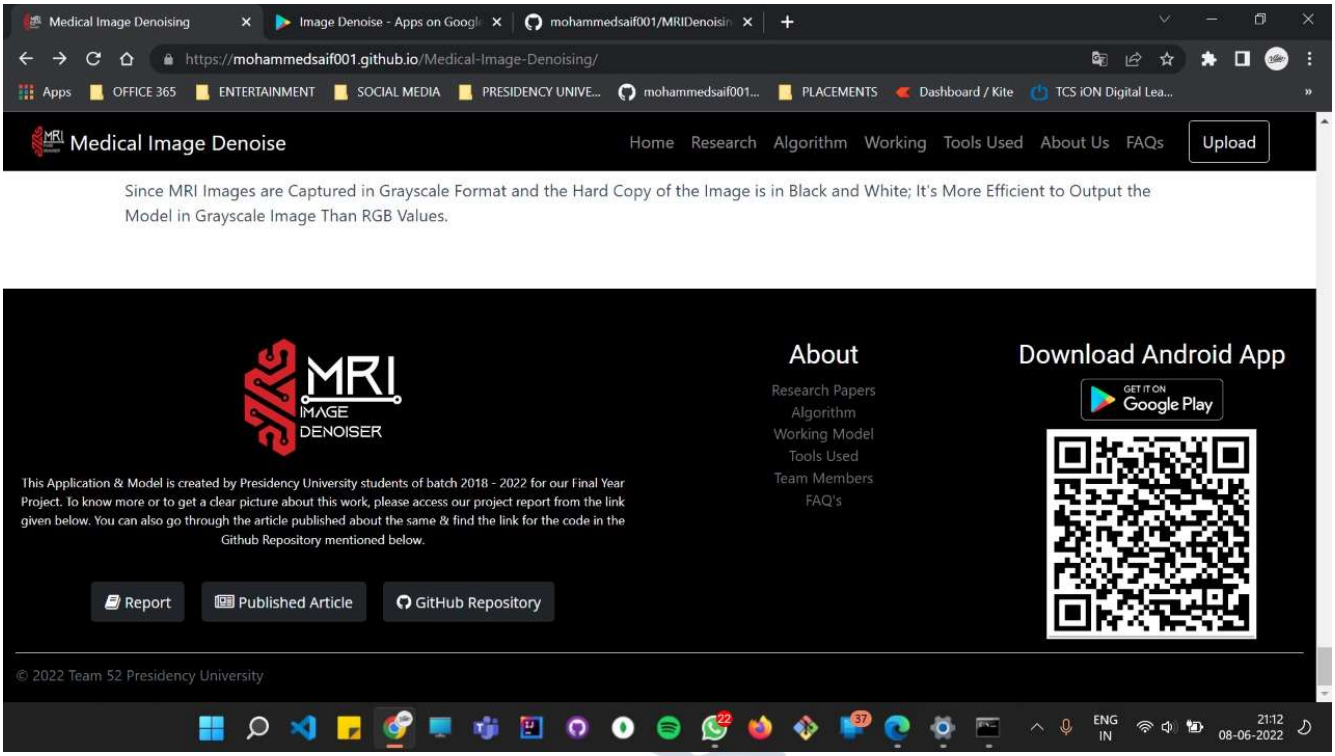

Medical Image Denoising
Image Denoise - Apps on Google
mohammedsaif001/MRIDenoising
https://mohammedsaif001.github.io/Medical-Image-Denoising/
Apps
OFFICE 365
ENTERTAINMENT
SOCIAL MEDIA
PRESIDENCY UNIVE...
mohammedsaif001...
PLACEMENTS
Dashboard / Kite
TCS iON Digital Lea...
Medical Image Denoise
Home
Research
Algorithm
Working
Tools Used
About Us
FAQs
Upload
Mohammed Saif
20181CSE0433
React JS Developer & UI/UX designing. As a Team Leader developing a strategic & systematic plan to accomplish predetermined tasks and ensuring that there is a collaborative work environment.
Syed Yunus
20181CSE0737
Backend Developer, created a backend on which the image processing model is embedded using flask and also created android application.
Mohammed Uvais
20181CSE0437
UI/UX designing ,Content Creation,Documentation and Website Testing
ENG
IN
21:12
08-06-2022

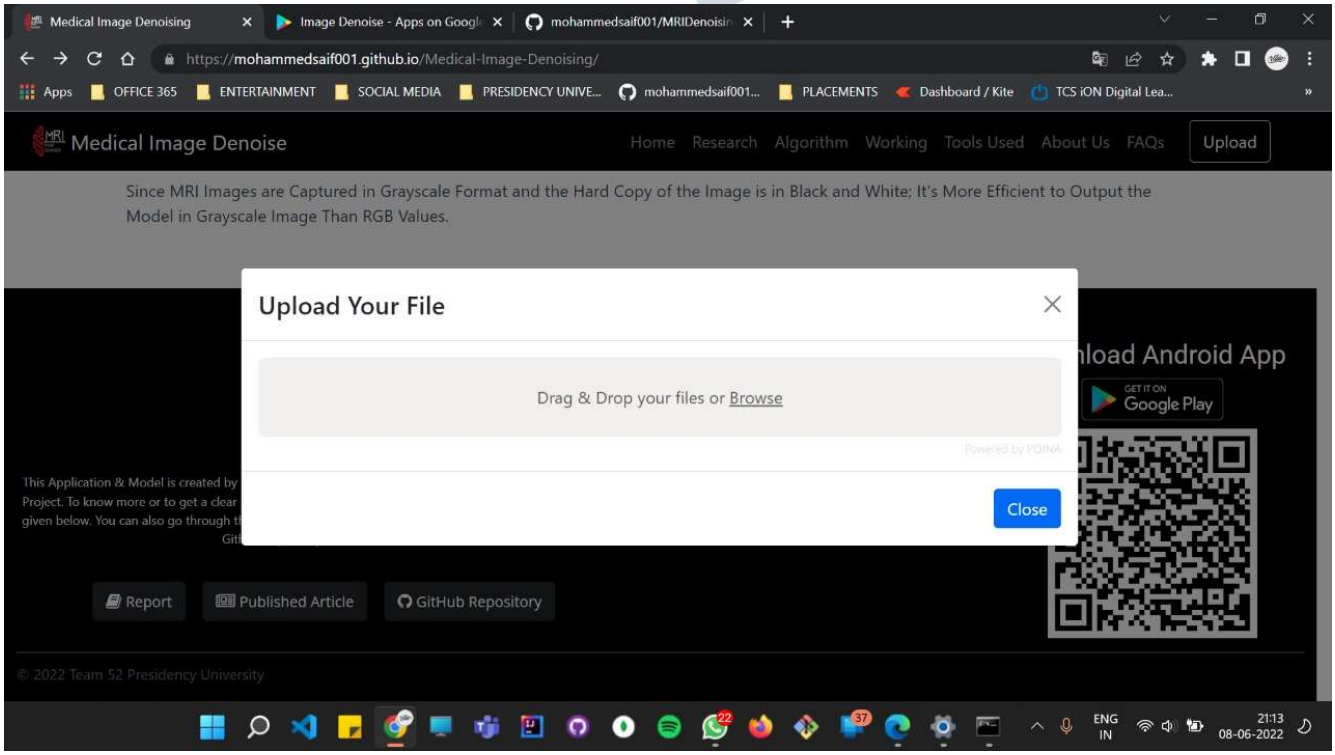

Medical Image Denoising
Image Denoise - Apps on Google
mohammedsaif001/MRIDenoising
https://mohammedsaif001.github.io/Medical-Image-Denoising/
Apps
OFFICE 365
ENTERTAINMENT
SOCIAL MEDIA
PRESIDENCY UNIVE...
mohammedsaif001...
PLACEMENTS
Dashboard / Kite
TCS iON Digital Lea...
Medical Image Denoise
Home
Research
Algorithm
Working
Tools Used
About Us
FAQs
Upload
Frequently Asked Questions
Is this Algorithm Applicable only for Medical Images?
Yes, This Algorithm is Applicable only for Medical Images. The reason being, our Model is trained to Remove Ration Noise Which is Present Mainly in MRI Images and hence our Model is Limited only to Medical Images .
Why Can't I Upload Regular Images to Denoise?
Why am I Unable to Upload Images in PNG/JPG/JPEG/SVG/GIF?
Why this Algorithm is Working only for Black & White or Grayscale images?
ENG
IN
21:12
08-06-2022

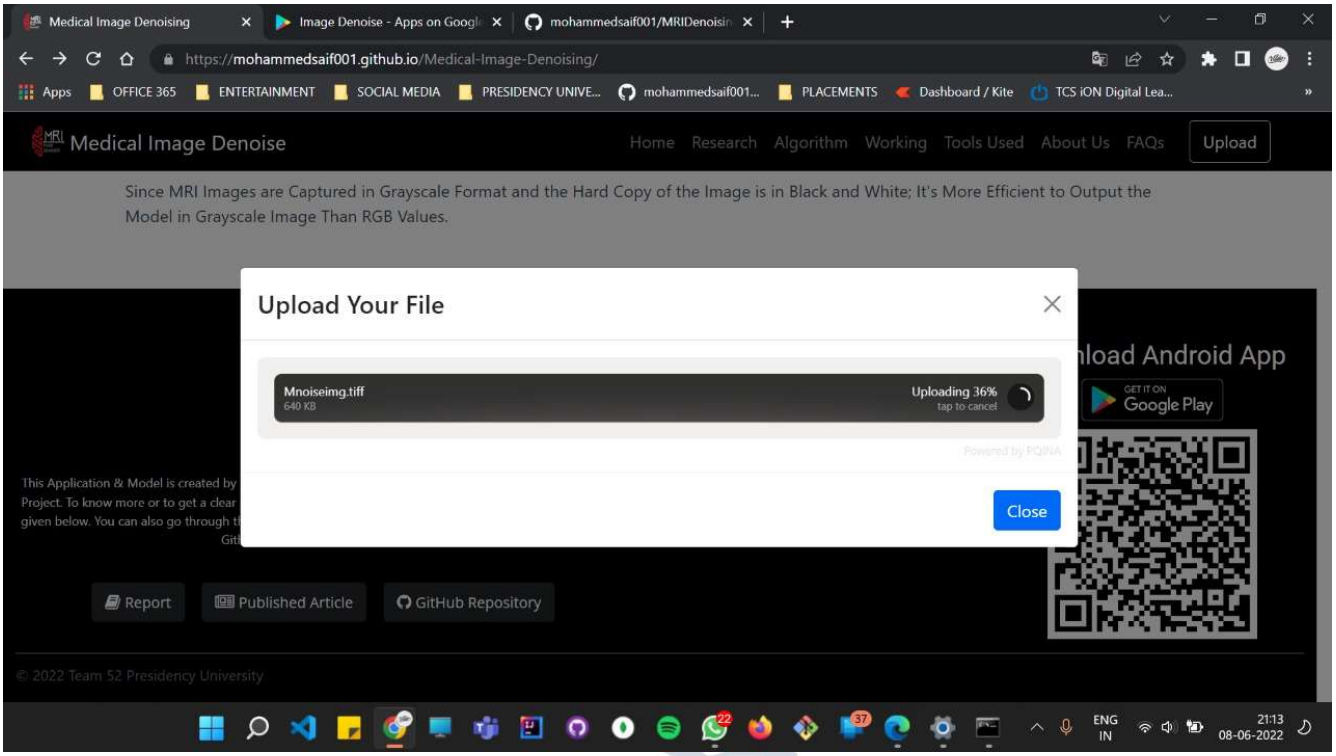

Medical Image Denoising
Image Denoise - Apps on Google
mohammedsaif001/MRIDenoising
https://mohammedsaif001.github.io/Medical-Image-Denoising/
Apps
OFFICE 365
ENTERTAINMENT
SOCIAL MEDIA
PRESIDENCY UNIVE...
mohammedsaif001...
PLACEMENTS
Dashboard / Kite
TCS iON Digital Lea...
Medical Image Denoise
Home
Research
Algorithm
Working
Tools Used
About Us
FAQs
Upload
Frequently Asked Questions
Is this Algorithm Applicable only for Medical Images?
Why Can't I Upload Regular Images to Denoise?
The Regular Images or Day-to-Day Images Contains Different Noises namely Gaussian Noise, Salt & Pepper Noise, Speckle Noise, Poisson Noise, Impulse Noise and much more but mostly not Ration noise. Whereas our Model is solely to denoise Ration Noise which is more applicable to Medical Images.
Why am I Unable to Upload Images in PNG/JPG/JPEG/SVG/GIF?
Why this Algorithm is Working only for Black & White or Grayscale images?
ENG
IN
21:12
08-06-2022

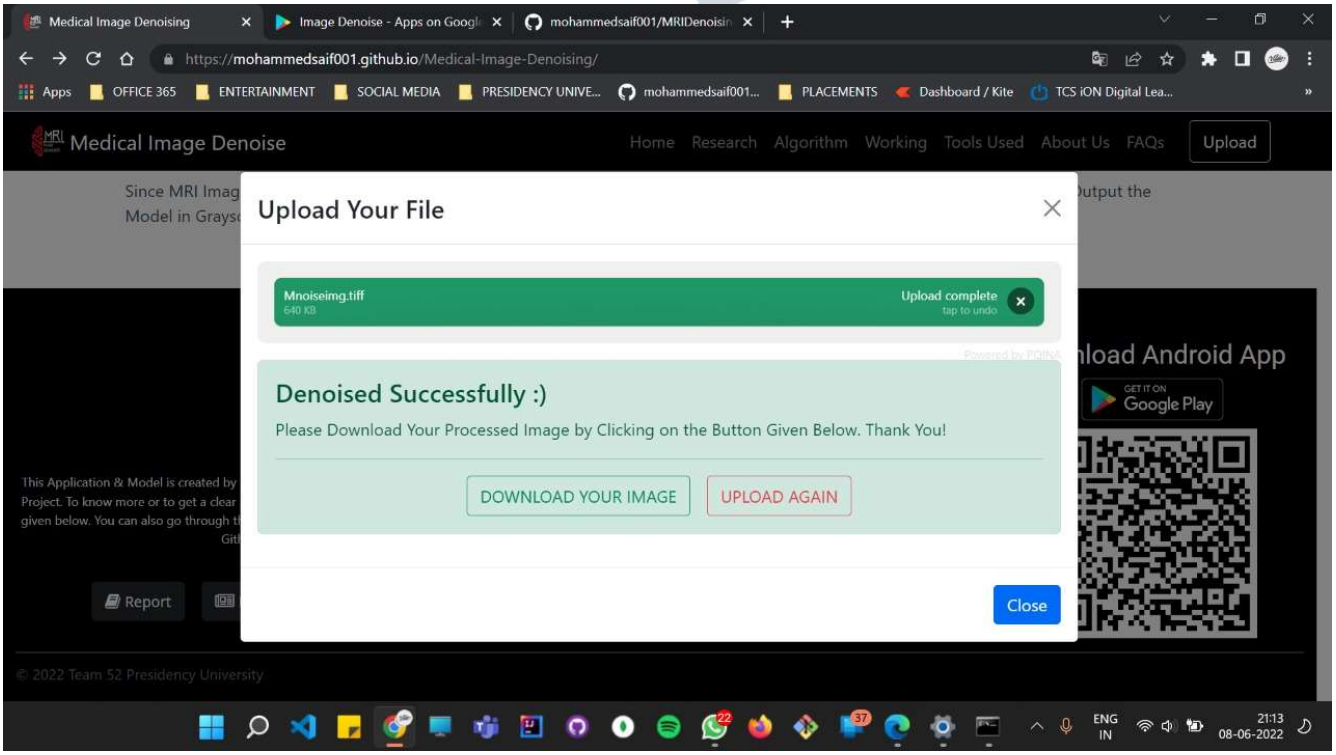

Medical Image Denoising
Image Denoise - Apps on Google
mohammedsaif001/MRIDenoising
https://mohammedsaif001.github.io/Medical-Image-Denoising/
Apps
OFFICE 365
ENTERTAINMENT
SOCIAL MEDIA
PRESIDENCY UNIVE...
mohammedsaif001...
PLACEMENTS
Dashboard / Kite
TCS iON Digital Lea...
Medical Image Denoise
Home
Research
Algorithm
Working
Tools Used
About Us
FAQs
Upload
Frequently Asked Questions
Is this Algorithm Applicable only for Medical Images?
Why Can't I Upload Regular Images to Denoise?
Why am I Unable to Upload Images in PNG/JPG/JPEG/SVG/GIF?
Our Model is Developed Stricly to Denoise Medical Images & Since Most MRI Images Format is TIFF/TIF and Not PNG/JPG/JPEG/SVG therefore our Model Doesn't Allow User to Upload These Format Images so that it Doesn't Spoil the Uploaded image.
Why this Algorithm is Working only for Black & White or Grayscale images?
ENG
IN
21:12
08-06-2022

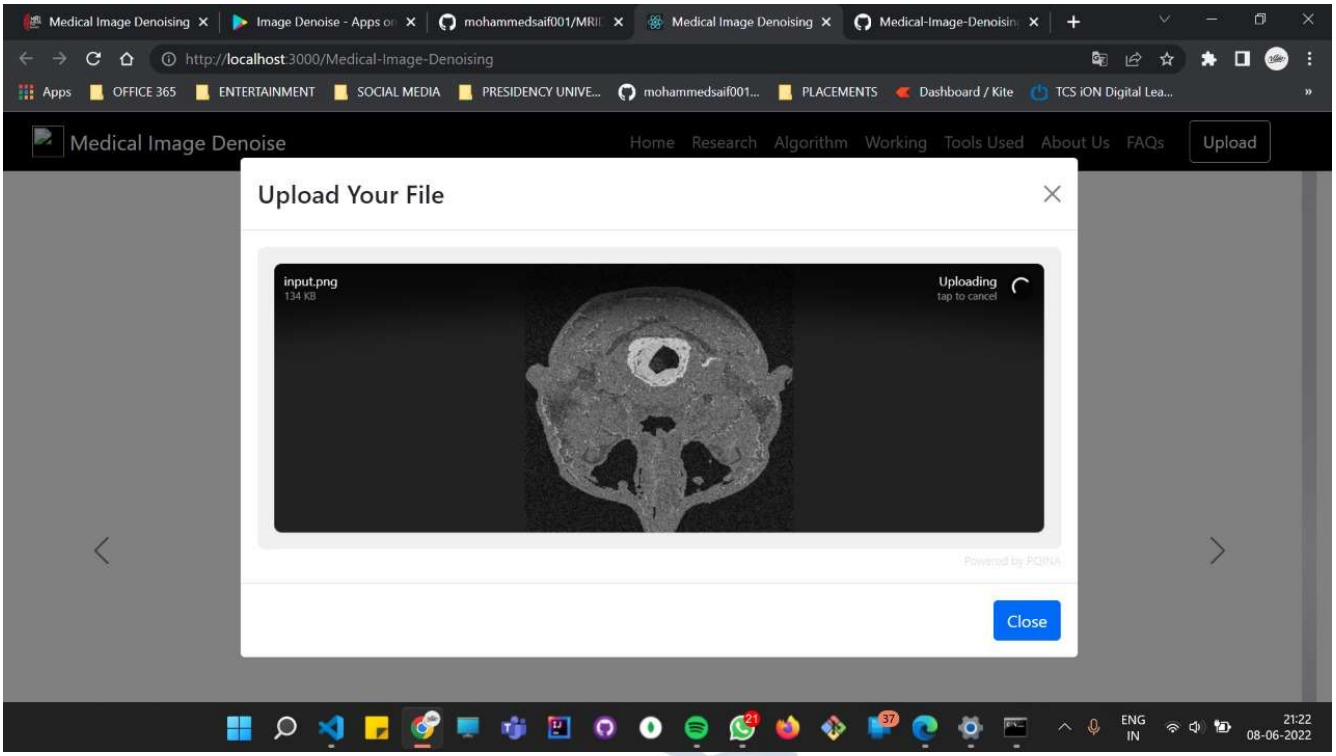

Medical Image Denoising
Image Denoise - Apps on Google
mohammedsaif001/MRIDenoising
https://mohammedsaif001.github.io/Medical-Image-Denoising/
Apps
OFFICE 365
ENTERTAINMENT
SOCIAL MEDIA
PRESIDENCY UNIVE...
mohammedsaif001...
PLACEMENTS
Dashboard / Kite
TCS iON Digital Lea...
Medical Image Denoise
Home
Research
Algorithm
Working
Tools Used
About Us
FAQs
Upload
Frequently Asked Questions
Is this Algorithm Applicable only for Medical Images?
Why Can't I Upload Regular Images to Denoise?
Why am I Unable to Upload Images in PNG/JPG/JPEG/SVG/GIF?
Why this Algorithm is Working only for Black & White or Grayscale images?
Since MRI Images are Captured in Grayscale Format and the Hard Copy of the Image is in Black and White; It's More Efficient to Output the Model in Grayscale Image Than RGB Values.
ENG
IN
21:12
08-06-2022

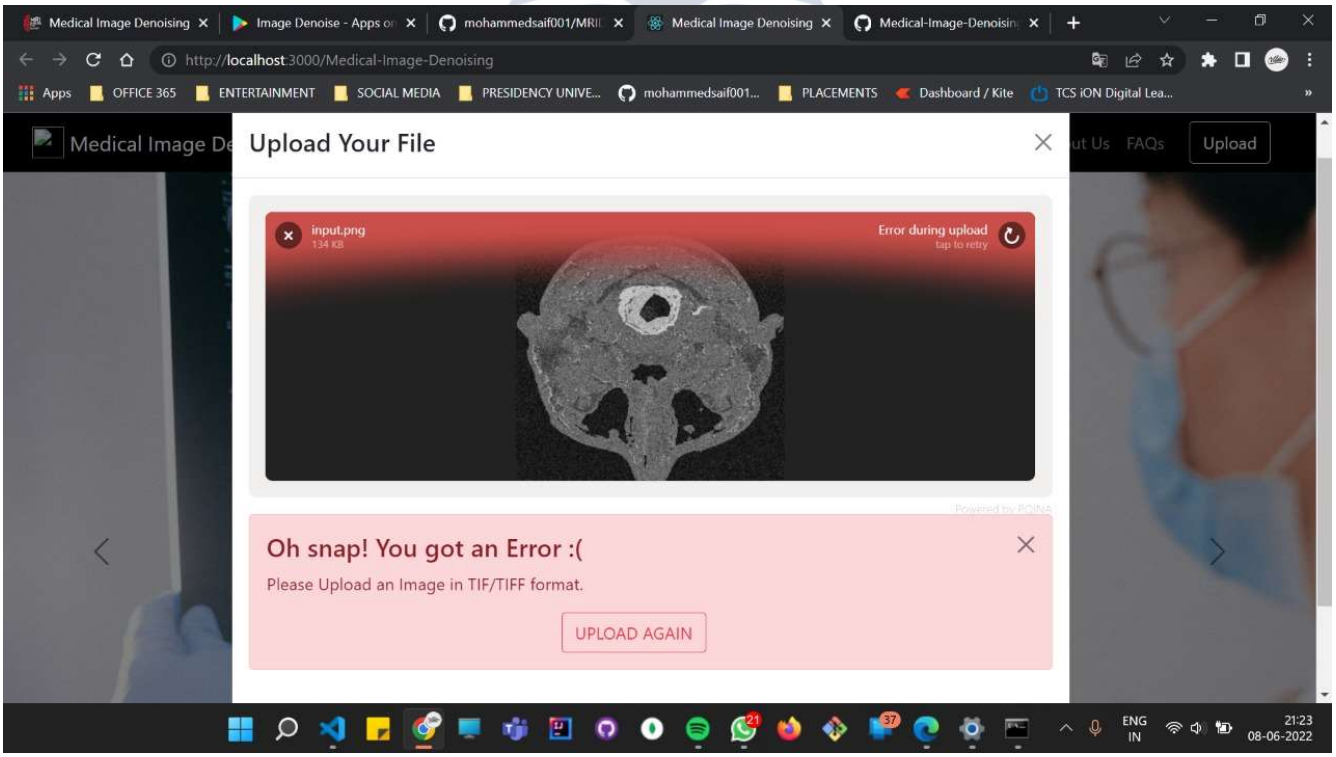

Medical Image Denoising
Image Denoise - Apps on Google
mohammedsaif001/MRIDenoising
https://mohammedsaif001.github.io/Medical-Image-Denoising/
Apps
OFFICE 365
ENTERTAINMENT
SOCIAL MEDIA
PRESIDENCY UNIVE...
mohammedsaif001...
PLACEMENTS
Dashboard / Kite
TCS iON Digital Lea...
Medical Image Denoise
Home
Research
Algorithm
Working
Tools Used
About Us
FAQs
Upload
Since MRI Images are Captured in Grayscale Format and the Hard Copy of the Image is in Black and White; It's More Efficient to Output the Model in Grayscale Image Than RGB Values.
MRI
IMAGE
DENOISER
This Application & Model is created by Presidency University students of batch 2018 - 2022 for our Final Year Project. To know more or to get a clear picture about this work, please access our project report from the link given below. You can also go through the article published about the same & find the link for the code in the Github Repository mentioned below.
Report
Published Article
GitHub Repository
About
Research Papers
Algorithm
Working Model
Tools Used
Team Members
FAQ's
Download Android App
GET IT ON Google Play
© 2022 Team 52 Presidency University
ENG
IN
21:12
08-06-2022


Medical Image Denoising
Image Denoise - Apps on Google
mohammedsaif001/MRIDenoising
https://mohammedsaif001.github.io/Medical-Image-Denoising/
Apps
OFFICE 365
ENTERTAINMENT
SOCIAL MEDIA
PRESIDENCY UNIVE...
mohammedsaif001...
PLACEMENTS
Dashboard / Kite
TCS iON Digital Lea...
Medical Image Denoise
Home
Research
Algorithm
Working
Tools Used
About Us
FAQs
Upload
Since MRI Images are Captured in Grayscale Format and the Hard Copy of the Image is in Black and White; It's More Efficient to Output the Model in Grayscale Image Than RGB Values.
Upload Your File
Drag & Drop your files or Browse
Close
Report
Published Article
GitHub Repository
© 2022 Team 52 Presidency University
ENG
IN
21:13
08-06-2022


Medical Image Denoising
Image Denoise - Apps on Google
mohammedsaif001/MRIDenoising
https://mohammedsaif001.github.io/Medical-Image-Denoising/
Medical Image Denoise
Home Research Algorithm Working Tools Used About Us FAQs Upload
Since MRI Images are Captured in Grayscale Format and the Hard Copy of the Image is in Black and White; It's More Efficient to Output the Model in Grayscale Image Than RGB Values.
Upload Your File
Mnoiseimg.tiff
640 KB
Uploading 36%
tap to cancel
Close
Report
Published Article
GitHub Repository
© 2022 Team 52 Presidency University


Medical Image Denoising
Image Denoise - Apps on Google
mohammedsaif001/MRIDenoising
https://mohammedsaif001.github.io/Medical-Image-Denoising/
Medical Image Denoise
Home Research Algorithm Working Tools Used About Us FAQs Upload
Upload Your File
Mnoiseimg.tiff
640 KB
Upload complete
tap to undo
Denoised Successfully :)
Please Download Your Processed Image by Clicking on the Button Given Below. Thank You!
DOWNLOAD YOUR IMAGE
UPLOAD AGAIN
Close
Report
© 2022 Team 52 Presidency University

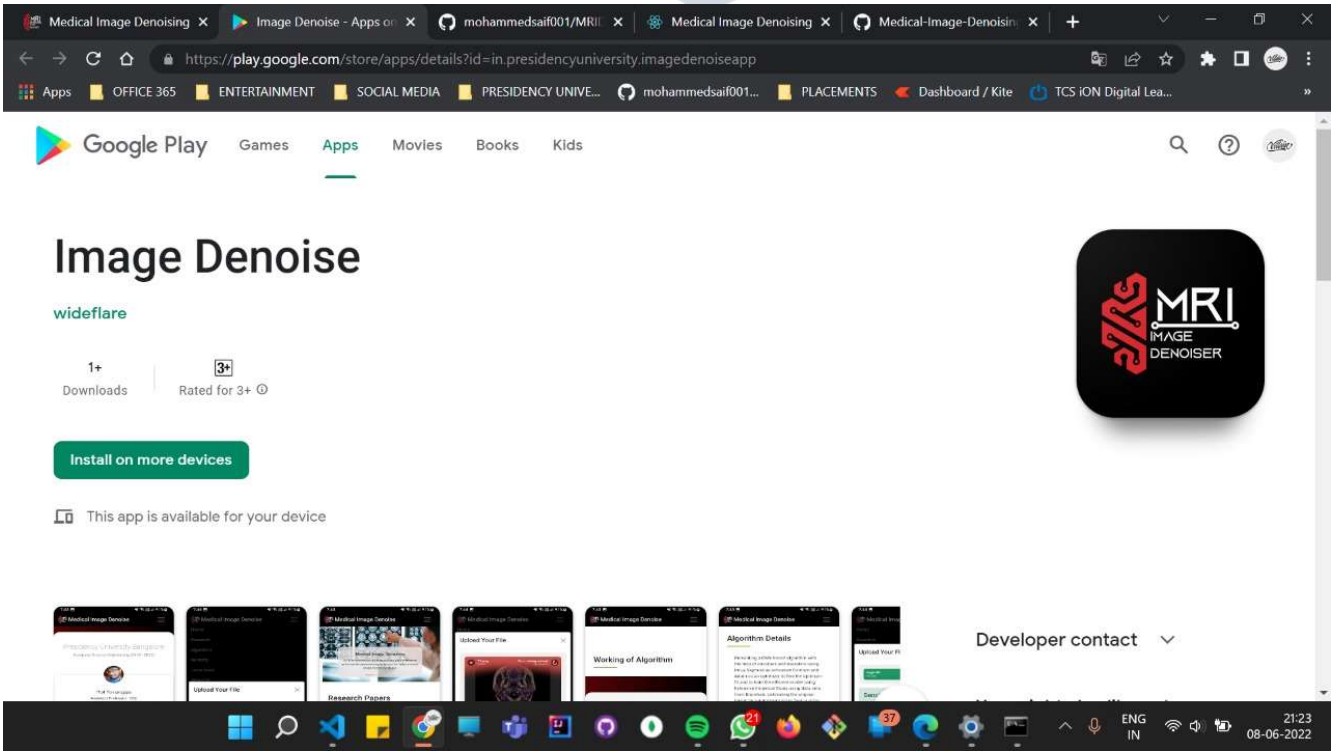

Medical Image Denoising
Image Denoise - Apps on
mohammedsaif001/MRII
Medical Image Denoising
Medical-Image-Denoisin
http://localhost:3000/Medical-Image-Denoising
Apps
OFFICE 365
ENTERTAINMENT
SOCIAL MEDIA
PRESIDENCY UNIVE...
mohammedsaif001...
PLACEMENTS
Dashboard / Kite
TCS iON Digital Lea...
Medical Image Denoise
Home
Research
Algorithm
Working
Tools Used
About Us
FAQs
Upload
Upload Your File
input.png
134 KB
Uploading
tap to cancel
Close
ENG
IN
21:22
08-06-2022

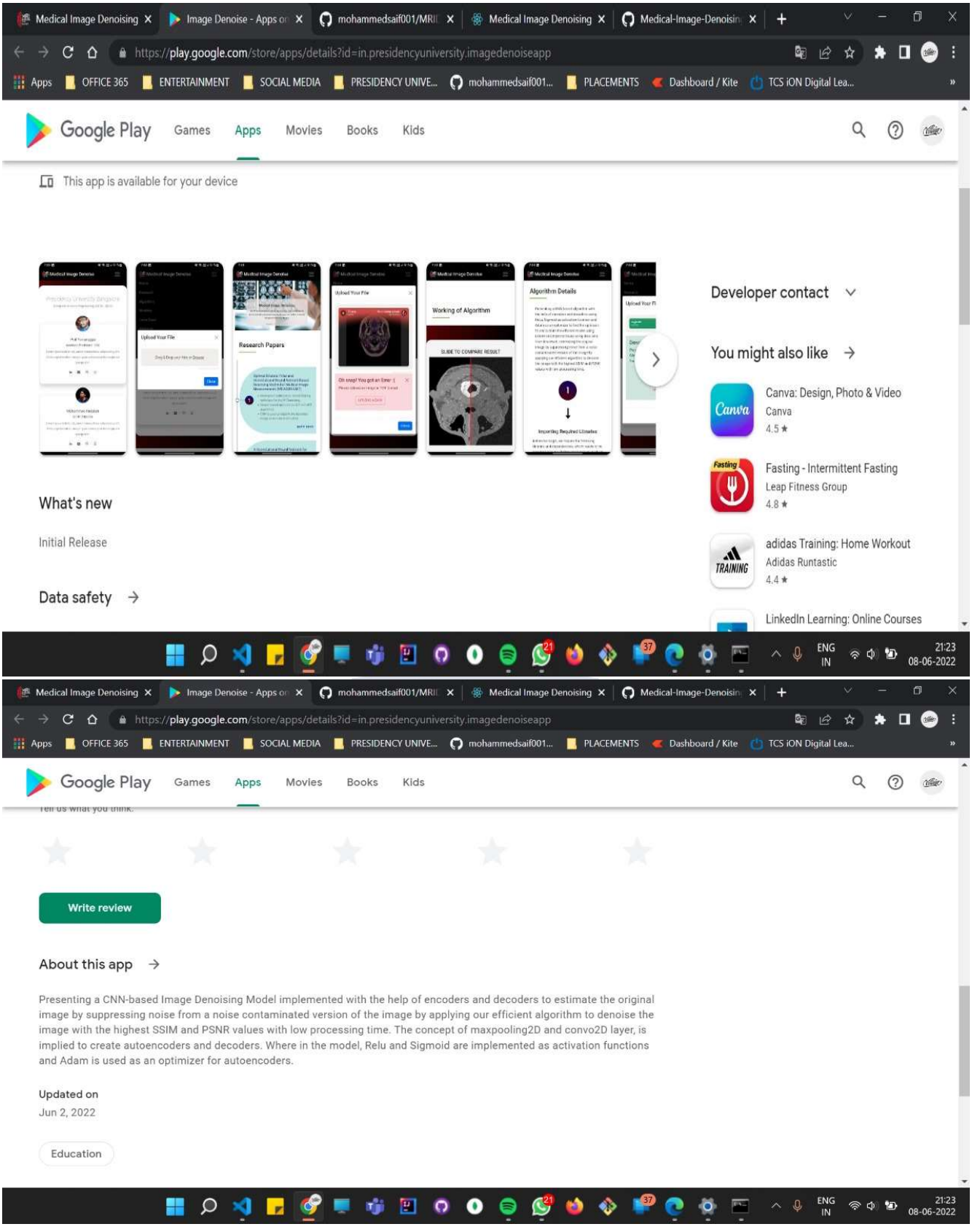

Medical Image Denoising
Image Denoise - Apps on
mohammedsaif001/MRII
Medical Image Denoising
Medical-Image-Denoisin
http://localhost:3000/Medical-Image-Denoising
Apps
OFFICE 365
ENTERTAINMENT
SOCIAL MEDIA
PRESIDENCY UNIVE...
mohammedsaif001...
PLACEMENTS
Dashboard / Kite
TCS iON Digital Lea...
Medical Image De
Upload Your File
FAQs
Upload
input.png
134 KB
Error during upload
tap to retry
Oh snap! You got an Error :(
Please Upload an Image in TIF/TIFF format.
UPLOAD AGAIN
ENG
IN
21:23
08-06-2022

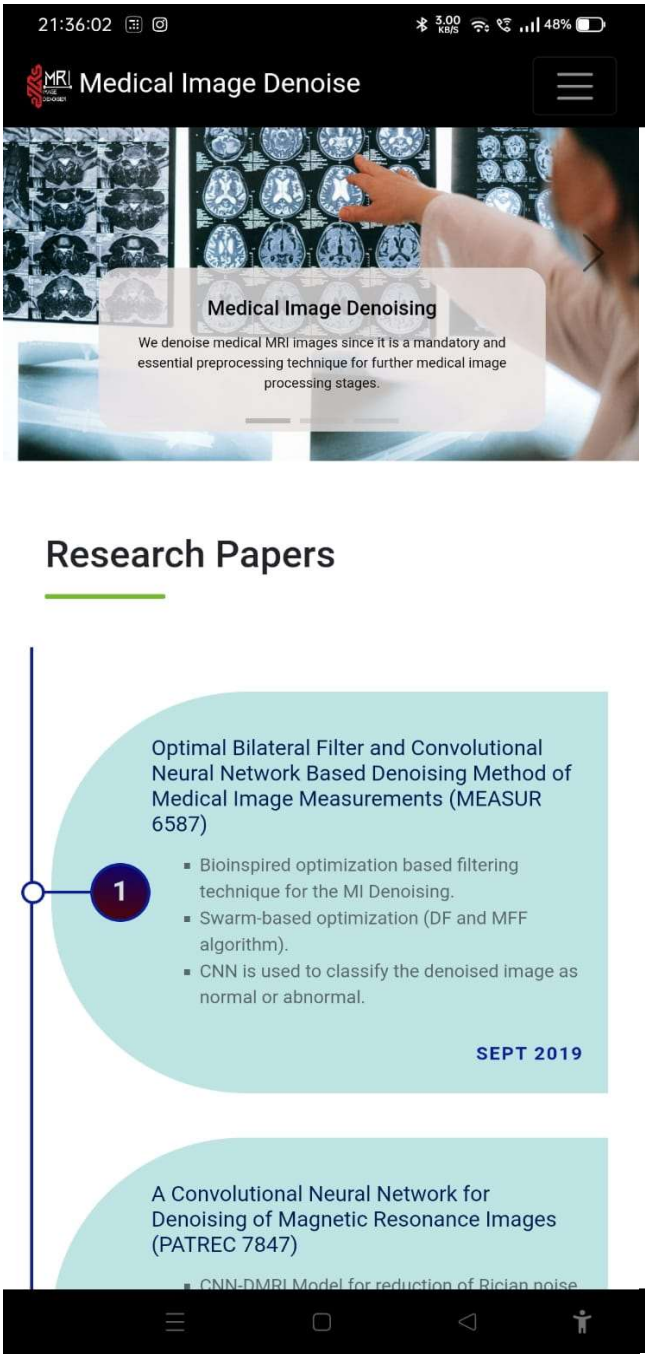
8.2 Image denoising Android App

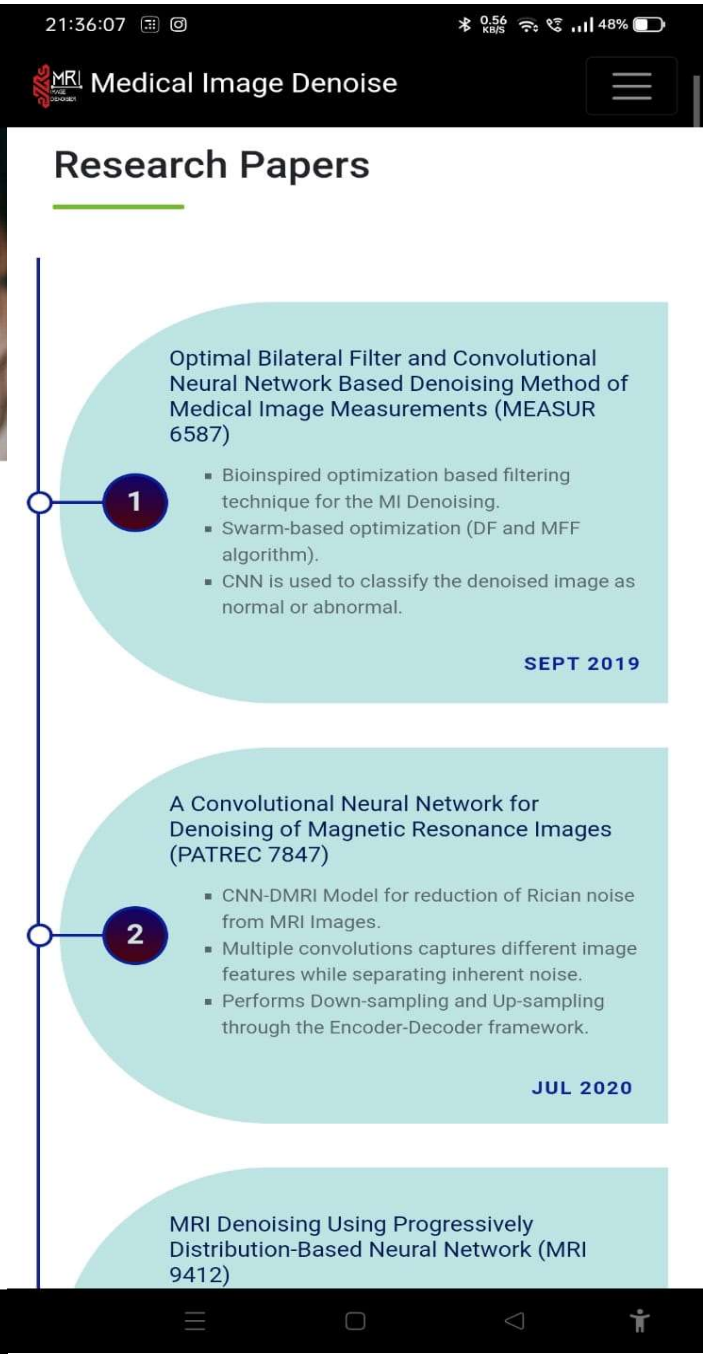
- **Link:** (PlayStore)
  https://play.google.com/store/apps/details?id=in.presidencyuniversity.imagedenoiseapp

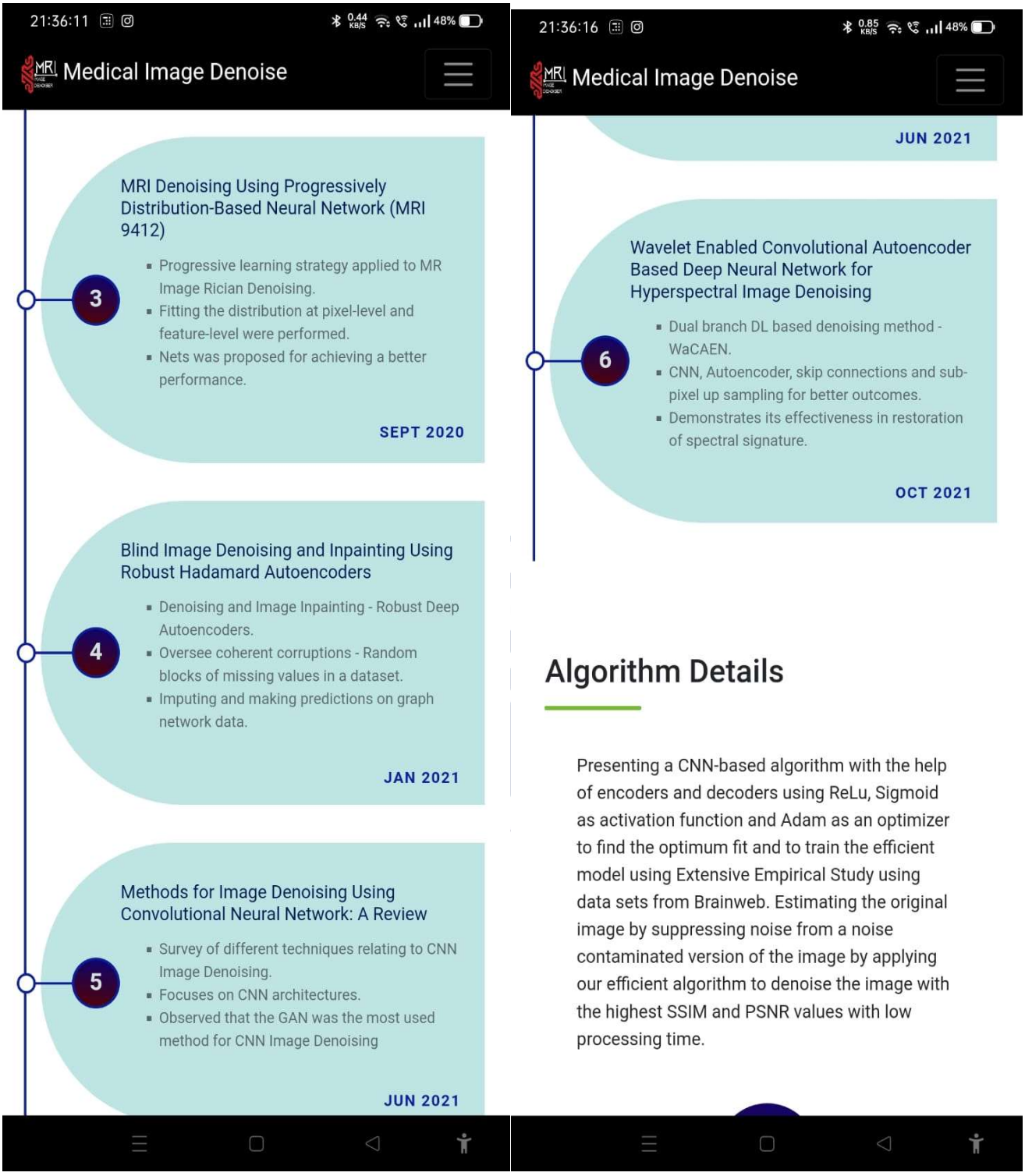
- **QR Code:**

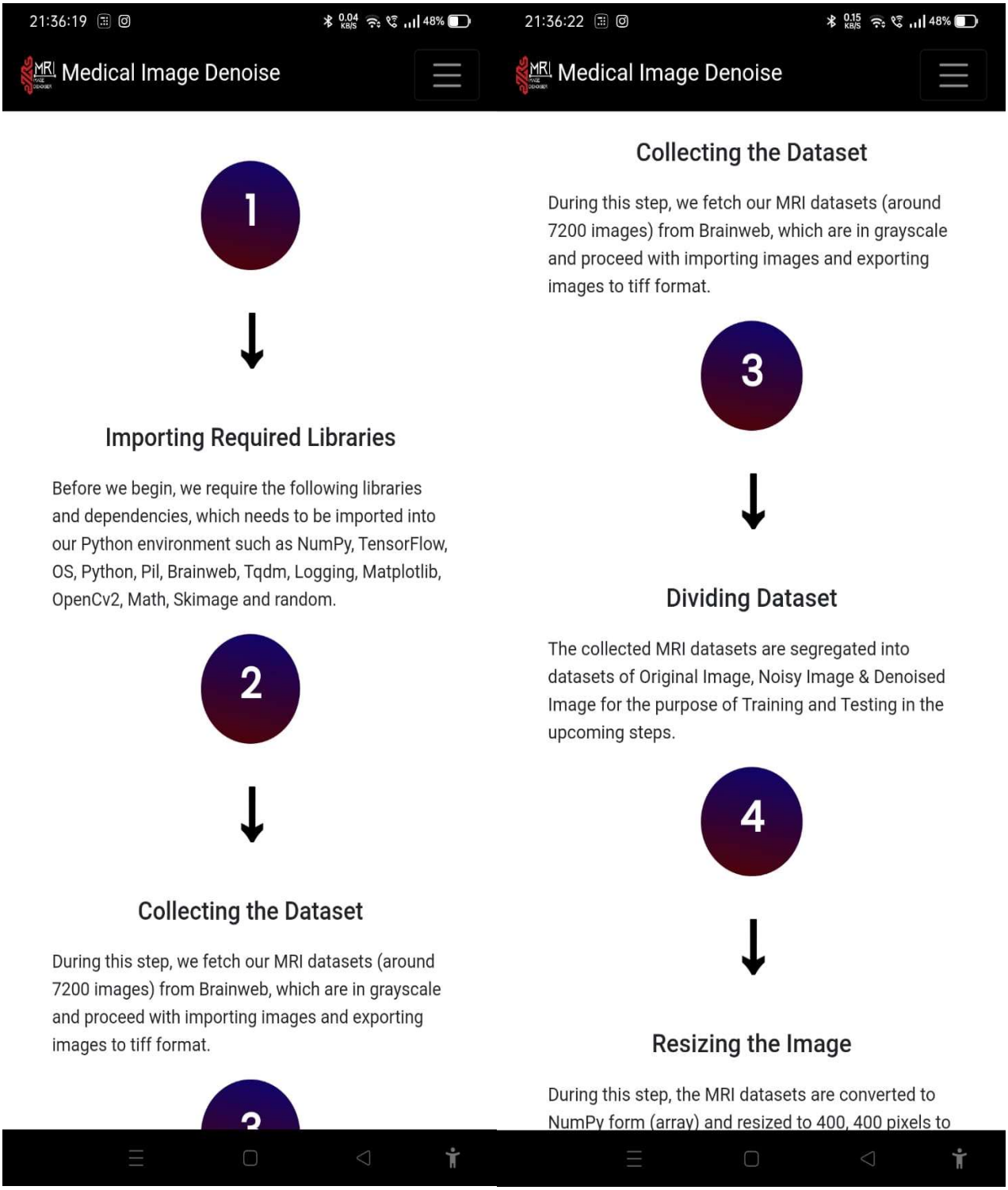


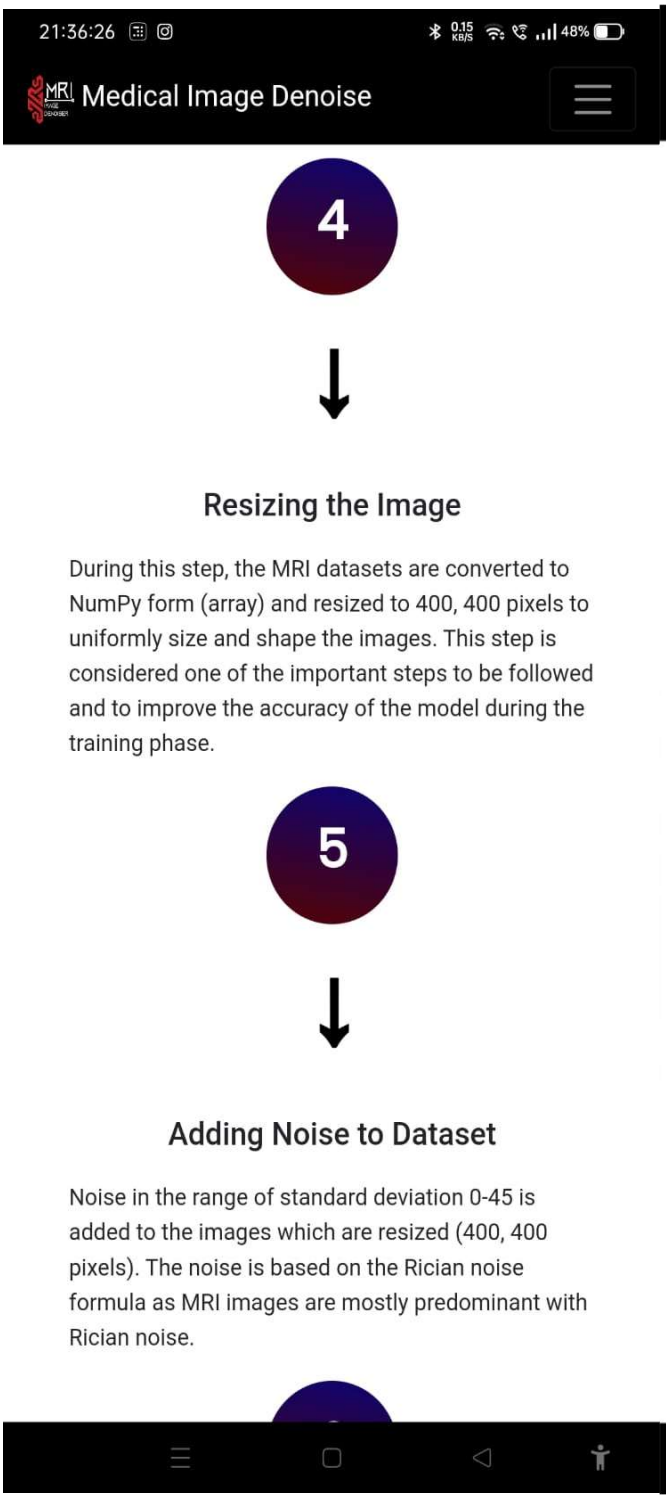

Medical Image Denoising
Image Denoise - Apps on
mohammedsaif001/MRII
Medical Image Denoising
Medical-Image-Denoisin
https://play.google.com/store/apps/details?id=in.presidencyuniversity.imagedenoiseapp
Apps
OFFICE 365
ENTERTAINMENT
SOCIAL MEDIA
PRESIDENCY UNIVE...
mohammedsaif001...
PLACEMENTS
Dashboard / Kite
TCS iON Digital Lea...
Google Play
Games
Apps
Movies
Books
Kids
This app is available for your device
Developer contact
You might also like
Canva: Design, Photo & Video
Canva
4.5
Fasting - Intermittent Fasting
Leap Fitness Group
4.8
adidas Training: Home Workout
Adidas Runtastic
4.4
LinkedIn Learning: Online Courses
What's new
Initial Release
Data safety
ENG
IN
21:23
08-06-2022
Write review
About this app
Presenting a CNN-based Image Denoising Model implemented with the help of encoders and decoders to estimate the original image by suppressing noise from a noise contaminated version of the image by applying our efficient algorithm to denoise the image with the highest SSIM and PSNR values with low processing time. The concept of maxpooling2D and convo2D layer, is implied to create autoencoders and decoders. Where in the model, Relu and Sigmoid are implemented as activation functions and Adam is used as an optimizer for autoencoders.
Updated on
Jun 2, 2022
Education

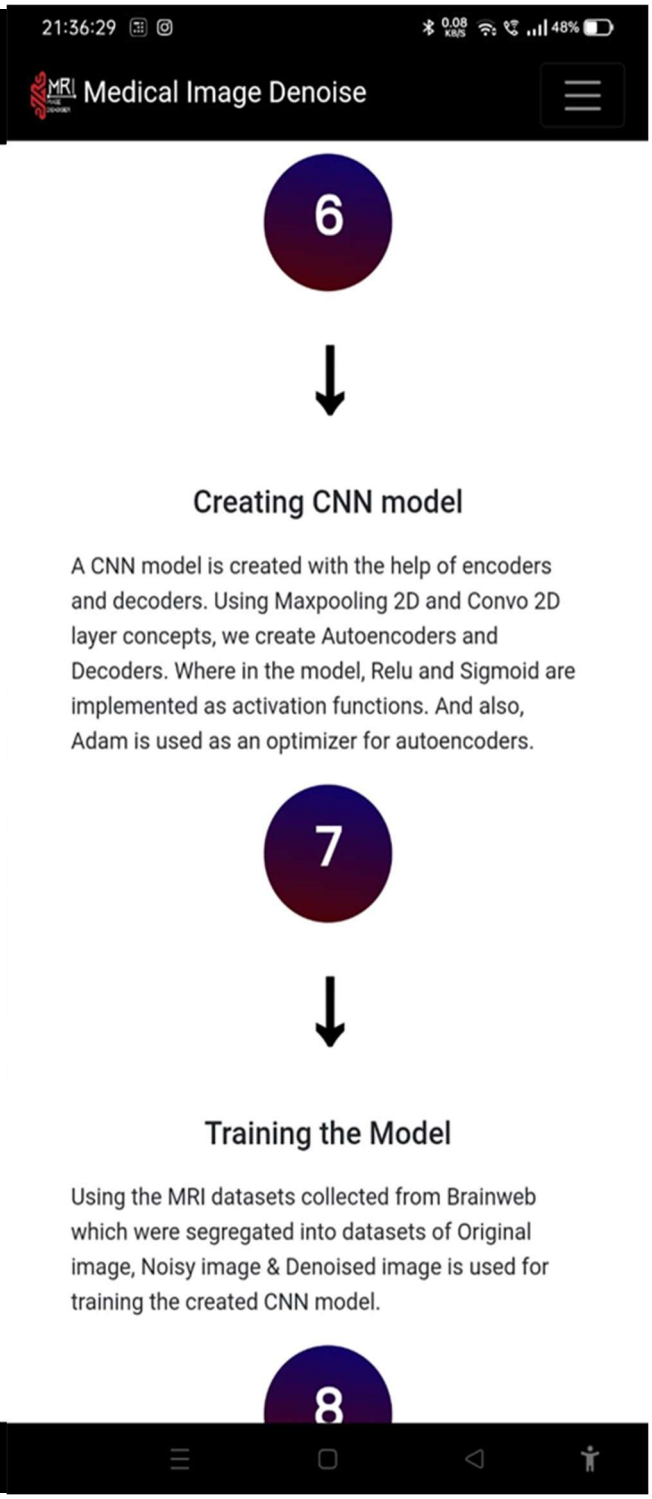

21:36:02
Medical Image Denoise
Medical Image Denoising
We denoise medical MRI images since it is a mandatory and essential preprocessing technique for further medical image processing stages.
Research Papers
Optimal Bilateral Filter and Convolutional Neural Network Based Denoising Method of Medical Image Measurements (MEASUR 6587)
1
Bioinspired optimization based filtering technique for the MI Denoising.
Swarm-based optimization (DF and MFF algorithm).
CNN is used to classify the denoised image as normal or abnormal.
SEPT 2019
A Convolutional Neural Network for Denoising of Magnetic Resonance Images (PATREC 7847)

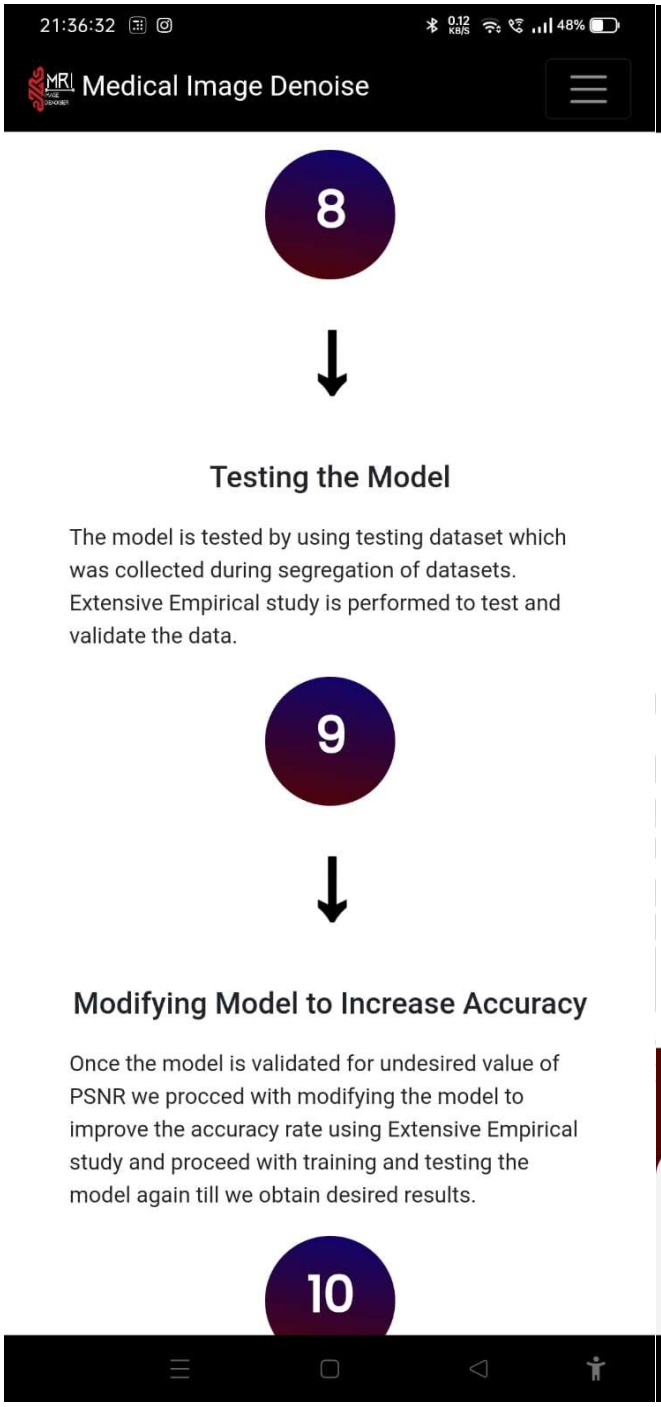

21:36:07
Medical Image Denoise
Research Papers
Optimal Bilateral Filter and Convolutional Neural Network Based Denoising Method of Medical Image Measurements (MEASUR 6587)
1
Bioinspired optimization based filtering technique for the MI Denoising.
Swarm-based optimization (DF and MFF algorithm).
CNN is used to classify the denoised image as normal or abnormal.
SEPT 2019
A Convolutional Neural Network for Denoising of Magnetic Resonance Images (PATREC 7847)
2
CNN-DMRI Model for reduction of Rician noise from MRI Images.
Multiple convolutions captures different image features while separating inherent noise.
Performs Down-sampling and Up-sampling through the Encoder-Decoder framework.
JUL 2020
MRI Denoising Using Progressively Distribution-Based Neural Network (MRI 9412)

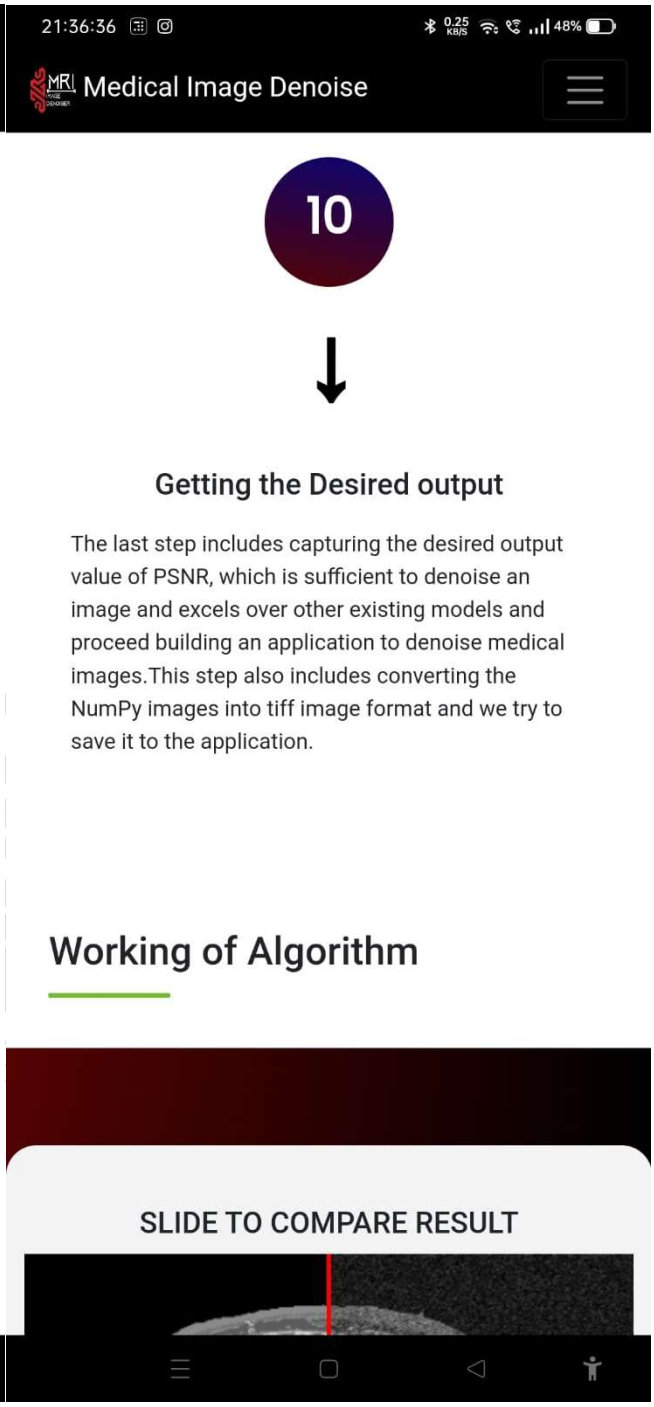

21:36:11
Medical Image Denoise
MRI Denoising Using Progressively Distribution-Based Neural Network (MRI 9412)
3
Progressive learning strategy applied to MR Image Rician Denoising.
Fitting the distribution at pixel-level and feature-level were performed.
Nets was proposed for achieving a better performance.
SEPT 2020
Blind Image Denoising and Inpainting Using Robust Hadamard Autoencoders
4
Denoising and Image Inpainting - Robust Deep Autoencoders.
Oversee coherent corruptions - Random blocks of missing values in a dataset.
Imputing and making predictions on graph network data.
JAN 2021
Methods for Image Denoising Using Convolutional Neural Network: A Review
5
Survey of different techniques relating to CNN Image Denoising.
Focuses on CNN architectures.
Observed that the GAN was the most used method for CNN Image Denoising
JUN 2021
21:36:16
Medical Image Denoise
JUN 2021
Wavelet Enabled Convolutional Autoencoder Based Deep Neural Network for Hyperspectral Image Denoising
6
Dual branch DL based denoising method - WaCAEN.
CNN, Autoencoder, skip connections and sub-pixel up sampling for better outcomes.
Demonstrates its effectiveness in restoration of spectral signature.
OCT 2021
Algorithm Details
Presenting a CNN-based algorithm with the help of encoders and decoders using ReLu, Sigmoid as activation function and Adam as an optimizer to find the optimum fit and to train the efficient model using Extensive Empirical Study using data sets from Brainweb. Estimating the original image by suppressing noise from a noise contaminated version of the image by applying our efficient algorithm to denoise the image with the highest SSIM and PSNR values with low processing time.

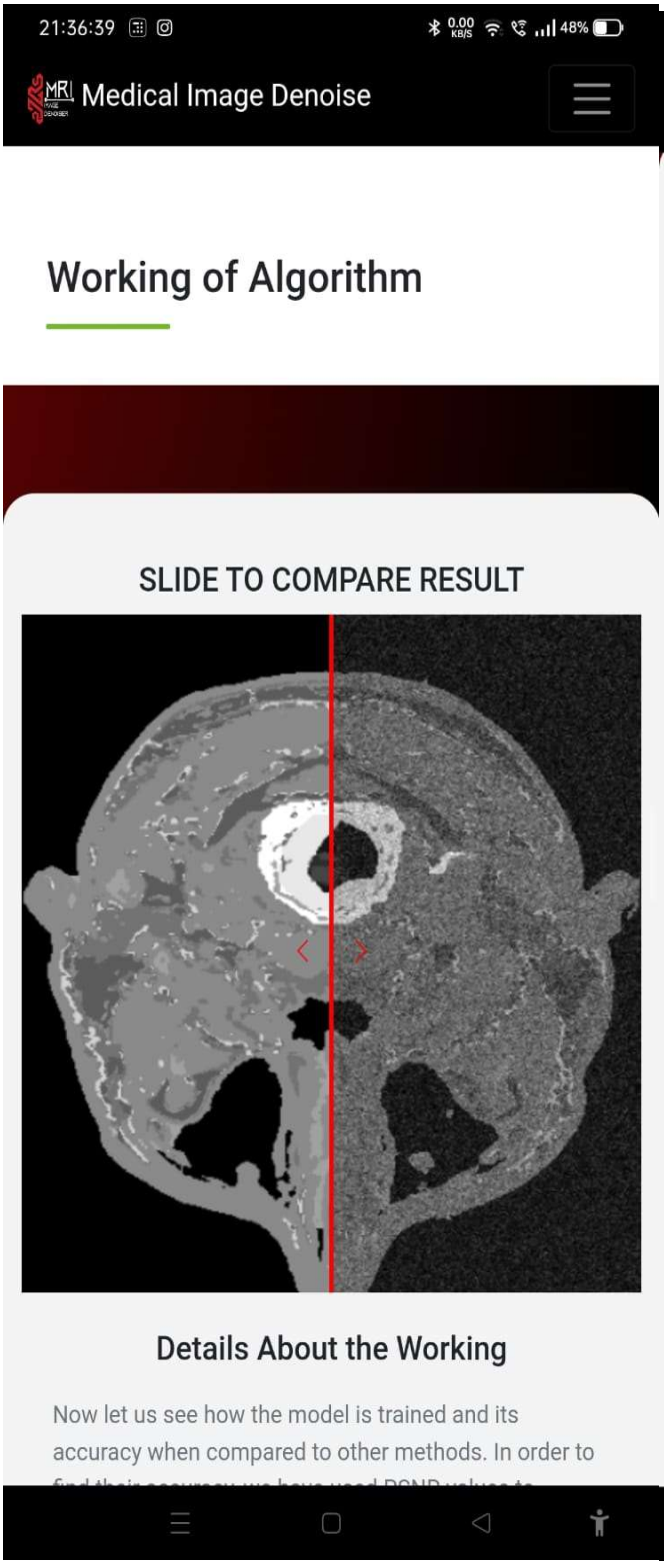

21:36:19
0.04 KB/S
48%
MRI Medical Image Denoise
1
Importing Required Libraries
Before we begin, we require the following libraries and dependencies, which needs to be imported into our Python environment such as NumPy, TensorFlow, OS, Python, Pil, Brainweb, Tqdm, Logging, Matplotlib, OpenCv2, Math, Skimage and random.
2
Collecting the Dataset
During this step, we fetch our MRI datasets (around 7200 images) from Brainweb, which are in grayscale and proceed with importing images and exporting images to tiff format.
21:36:22
0.15 KB/S
48%
MRI Medical Image Denoise
Collecting the Dataset
During this step, we fetch our MRI datasets (around 7200 images) from Brainweb, which are in grayscale and proceed with importing images and exporting images to tiff format.
3
Dividing Dataset
The collected MRI datasets are segregated into datasets of Original Image, Noisy Image & Denoised Image for the purpose of Training and Testing in the upcoming steps.
4
Resizing the Image
During this step, the MRI datasets are converted to NumPy form (array) and resized to 400, 400 pixels to

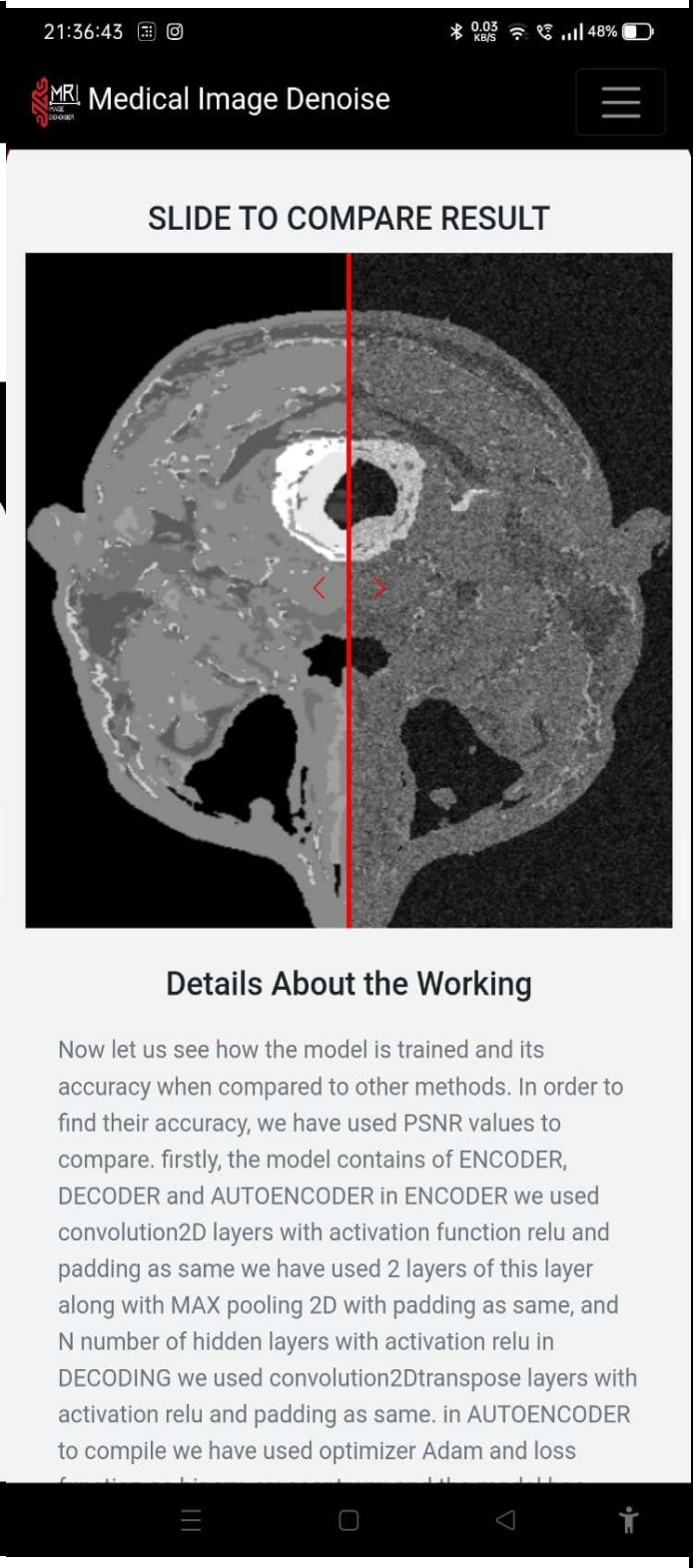

21:36:26
0.15 KB/S
48%
MRI Medical Image Denoise
4
Resizing the Image
During this step, the MRI datasets are converted to NumPy form (array) and resized to 400, 400 pixels to uniformly size and shape the images. This step is considered one of the important steps to be followed and to improve the accuracy of the model during the training phase.
5
Adding Noise to Dataset
Noise in the range of standard deviation 0-45 is added to the images which are resized (400, 400 pixels). The noise is based on the Rician noise formula as MRI images are mostly predominant with Rician noise.

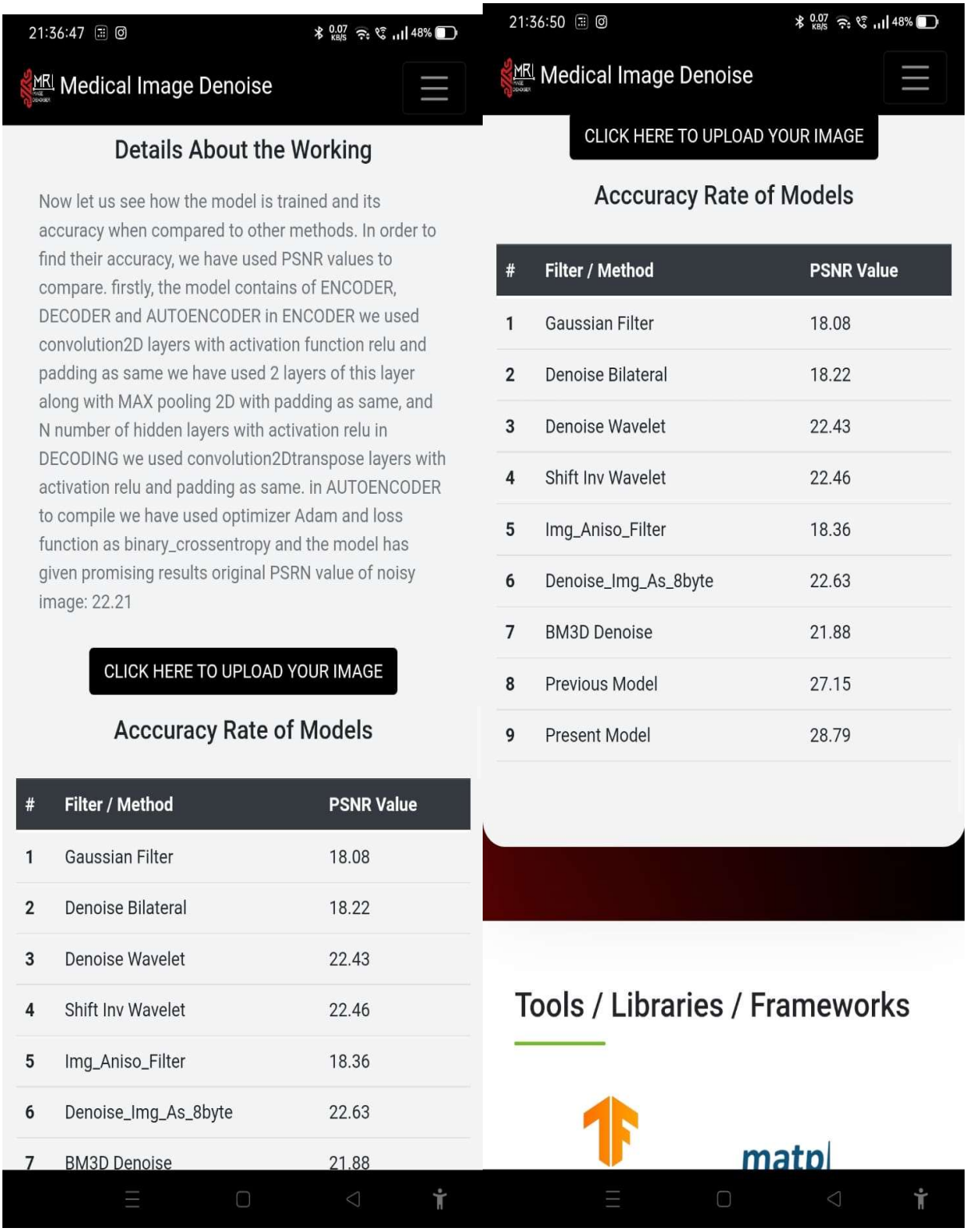

21:36:29
0.08 KB/S
48%
MRI Medical Image Denoise
6
Creating CNN model
A CNN model is created with the help of encoders and decoders. Using Maxpooling 2D and Convo 2D layer concepts, we create Autoencoders and Decoders. Where in the model, Relu and Sigmoid are implemented as activation functions. And also, Adam is used as an optimizer for autoencoders.
7
Training the Model
Using the MRI datasets collected from Brainweb which were segregated into datasets of Original image, Noisy image & Denoised image is used for training the created CNN model.
8

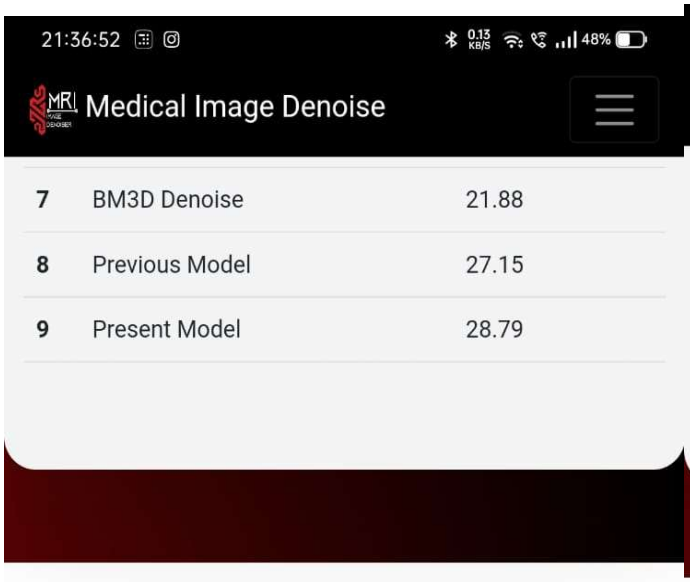

21:36:32
Medical Image Denoise
8
Testing the Model
The model is tested by using testing dataset which was collected during segregation of datasets. Extensive Empirical study is performed to test and validate the data.
9
Modifying Model to Increase Accuracy
Once the model is validated for undesired value of PSNR we procced with modifying the model to improve the accuracy rate using Extensive Empirical study and proceed with training and testing the model again till we obtain desired results.
10

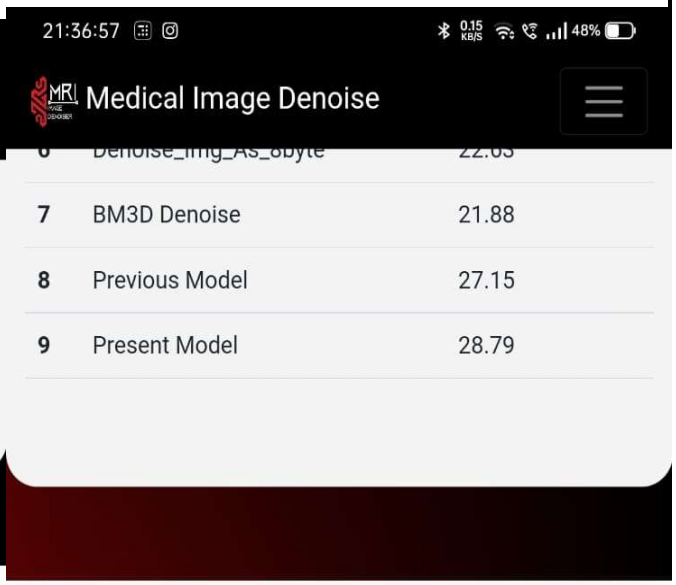

21:36:36
Medical Image Denoise
10
Getting the Desired output
The last step includes capturing the desired output value of PSNR, which is sufficient to denoise an image and excels over other existing models and proceed building an application to denoise medical images.This step also includes converting the NumPy images into tiff image format and we try to save it to the application.
Working of Algorithm
SLIDE TO COMPARE RESULT


21:36:39
0.00 KB/S
48%
Medical Image Denoise
Working of Algorithm
SLIDE TO COMPARE RESULT
Details About the Working
Now let us see how the model is trained and its accuracy when compared to other methods. In order to


21:36:43
0.03 KB/S
48%
Medical Image Denoise
SLIDE TO COMPARE RESULT
Details About the Working
Now let us see how the model is trained and its accuracy when compared to other methods. In order to find their accuracy, we have used PSNR values to compare. firstly, the model contains of ENCODER, DECODER and AUTOENCODER in ENCODER we used convolution2D layers with activation function relu and padding as same we have used 2 layers of this layer along with MAX pooling 2D with padding as same, and N number of hidden layers with activation relu in DECODING we used convolution2Dtranspose layers with activation relu and padding as same. in AUTOENCODER to compile we have used optimizer Adam and loss


21:36:47
Medical Image Denoise
Details About the Working
Now let us see how the model is trained and its accuracy when compared to other methods. In order to find their accuracy, we have used PSNR values to compare. firstly, the model contains of ENCODER, DECODER and AUTOENCODER in ENCODER we used convolution2D layers with activation function relu and padding as same we have used 2 layers of this layer along with MAX pooling 2D with padding as same, and N number of hidden layers with activation relu in DECODING we used convolution2Dtranspose layers with activation relu and padding as same. in AUTOENCODER to compile we have used optimizer Adam and loss function as binary_crossentropy and the model has given promising results original PSRN value of noisy image: 22.21
CLICK HERE TO UPLOAD YOUR IMAGE
Acccuracy Rate of Models
Filter / Method PSNR Value
1 Gaussian Filter 18.08
2 Denoise Bilateral 18.22
3 Denoise Wavelet 22.43
4 Shift Inv Wavelet 22.46
5 Img_Aniso_Filter 18.36
6 Denoise_Img_As_8byte 22.63
7 BM3D Denoise 21.88
21:36:50
Medical Image Denoise
CLICK HERE TO UPLOAD YOUR IMAGE
Acccuracy Rate of Models
Filter / Method PSNR Value
1 Gaussian Filter 18.08
2 Denoise Bilateral 18.22
3 Denoise Wavelet 22.43
4 Shift Inv Wavelet 22.46
5 Img_Aniso_Filter 18.36
6 Denoise_Img_As_8byte 22.63
7 BM3D Denoise 21.88
8 Previous Model 27.15
9 Present Model 28.79
Tools / Libraries / Frameworks





Tools / Libraries / Frameworks

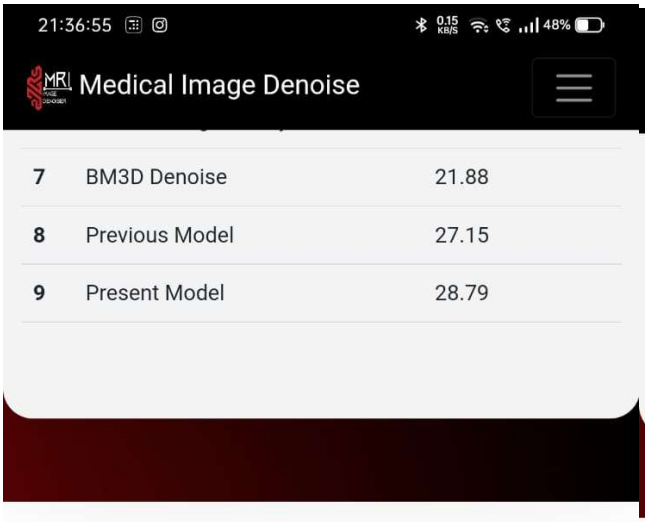
Keras kaggle

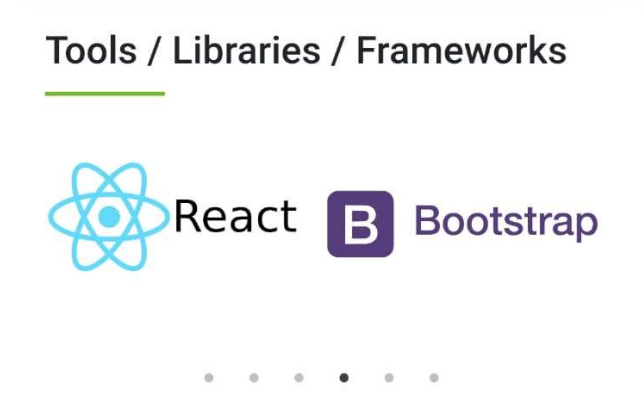
Tools / Libraries / Frameworks

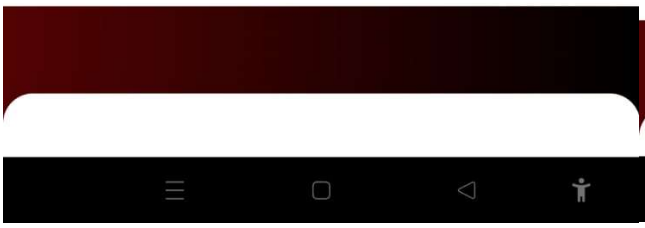
JS jQuery

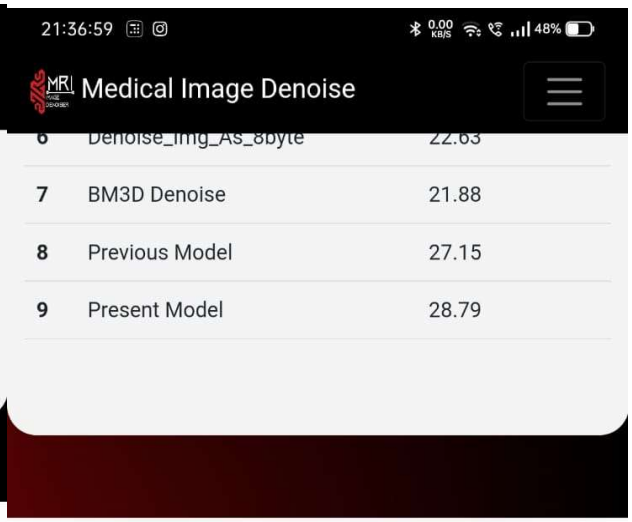
Team Member Details

Team Member Details

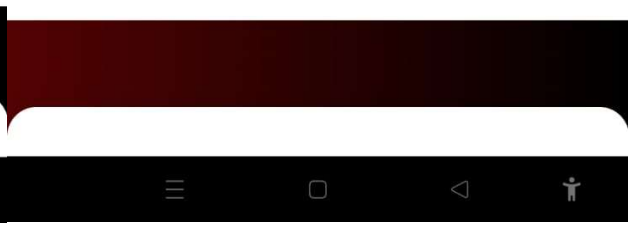

21:36:55
Medical Image Denoise
7 BM3D Denoise 21.88
8 Previous Model 27.15
9 Present Model 28.79

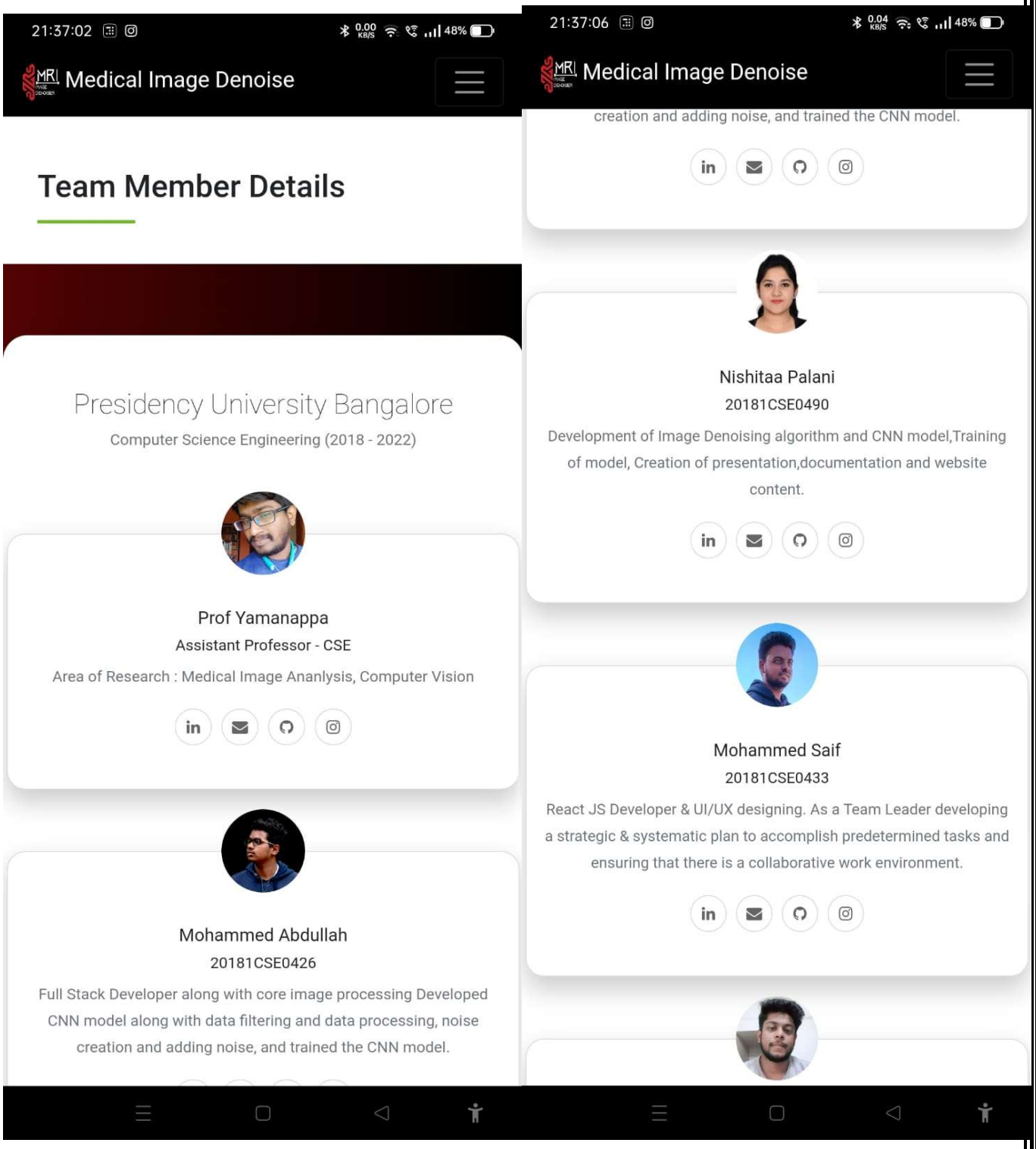

Tools / Libraries / Frameworks
React
Bootstrap

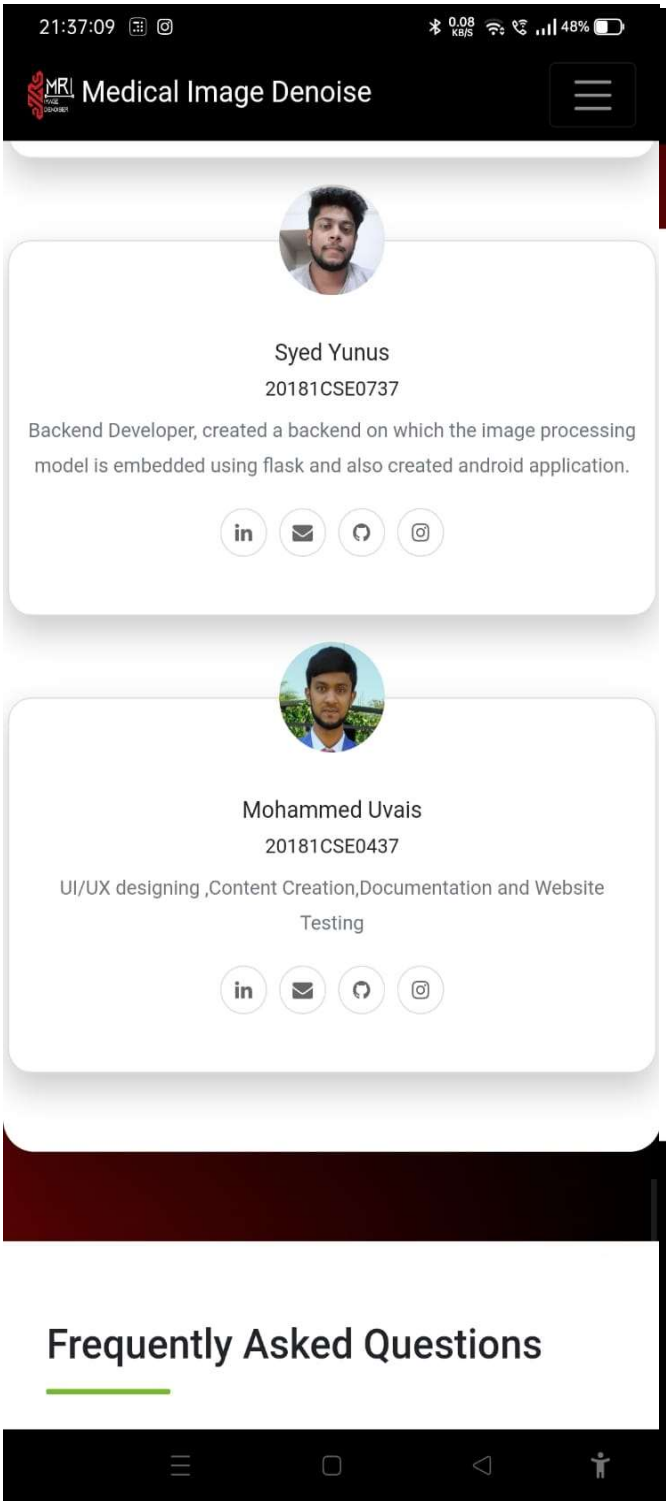

Team Member Details

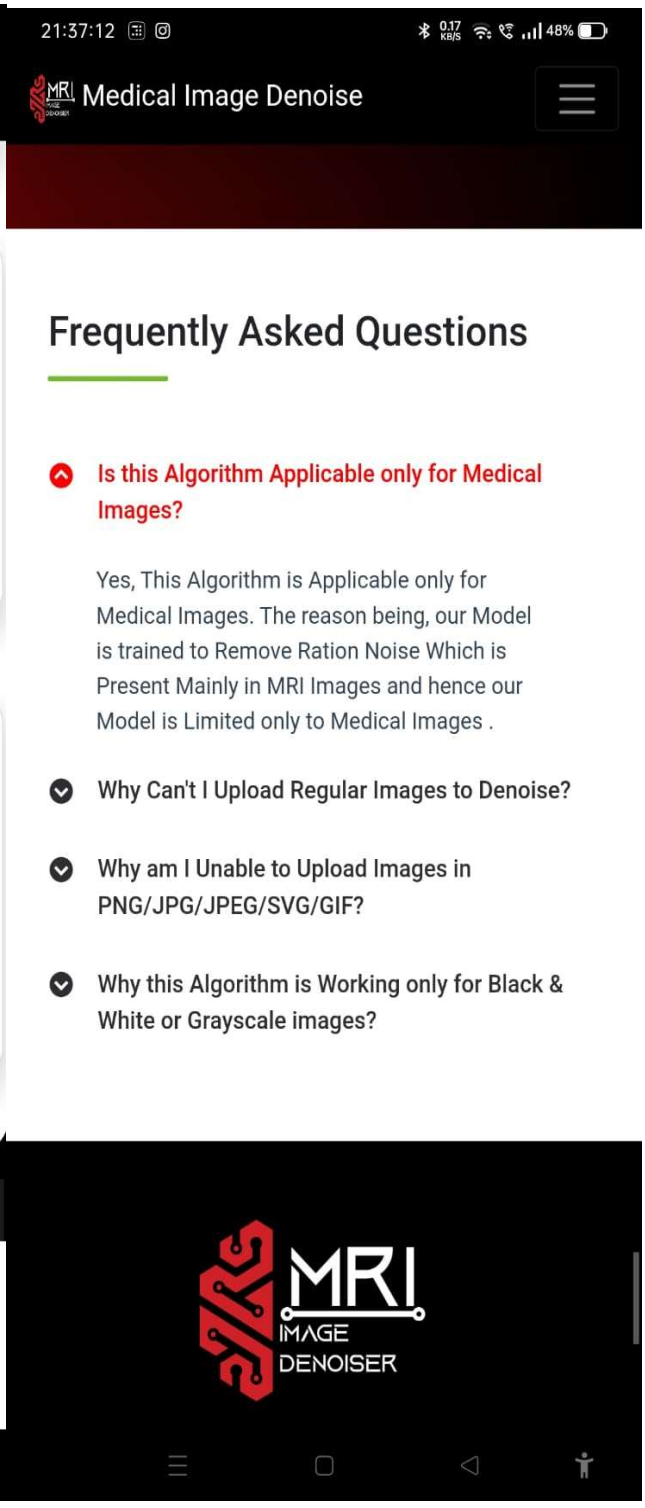

21:36:59
Medical Image Denoise
7 BM3D Denoise 21.88
8 Previous Model 27.15
9 Present Model 28.79

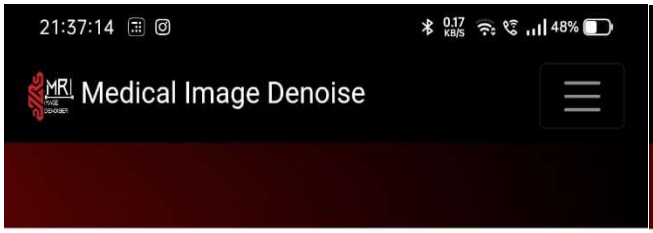

Tools / Libraries / Frameworks
aws
Flask

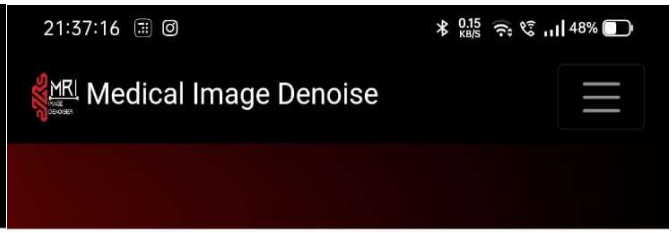

Team Member Details


21:37:02
Medical Image Denoise
Team Member Details
Presidency University Bangalore
Computer Science Engineering (2018 - 2022)
Prof Yamanappa
Assistant Professor - CSE
Area of Research : Medical Image Ananlysis, Computer Vision
Mohammed Abdullah
20181CSE0426
Full Stack Developer along with core image processing Developed CNN model along with data filtering and data processing, noise creation and adding noise, and trained the CNN model.
21:37:06
Medical Image Denoise
creation and adding noise, and trained the CNN model.
Nishitaa Palani
20181CSE0490
Development of Image Denoising algorithm and CNN model,Training of model, Creation of presentation,documentation and website content.
Mohammed Saif
20181CSE0433
React JS Developer & UI/UX designing. As a Team Leader developing a strategic & systematic plan to accomplish predetermined tasks and ensuring that there is a collaborative work environment.


21:37:09
Medical Image Denoise
Syed Yunus
20181CSE0737
Backend Developer, created a backend on which the image processing model is embedded using flask and also created android application.
Mohammed Uvais
20181CSE0437
UI/UX designing ,Content Creation,Documentation and Website Testing
Frequently Asked Questions


21:37:12
Medical Image Denoise
Frequently Asked Questions
Is this Algorithm Applicable only for Medical Images?
Yes, This Algorithm is Applicable only for Medical Images. The reason being, our Model is trained to Remove Ration Noise Which is Present Mainly in MRI Images and hence our Model is Limited only to Medical Images .
Why Can't I Upload Regular Images to Denoise?
Why am I Unable to Upload Images in PNG/JPG/JPEG/SVG/GIF?
Why this Algorithm is Working only for Black & White or Grayscale images?
MRI
IMAGE
DENOISER





Frequently Asked Questions

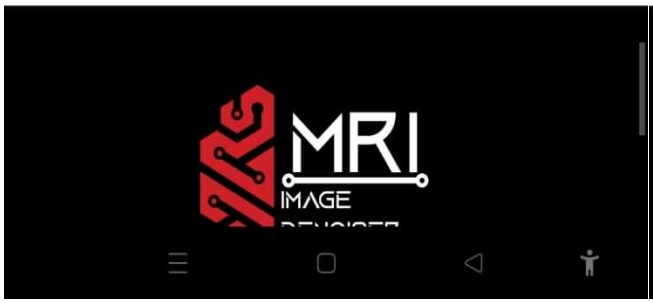
- Is this Algorithm Applicable only for Medical Images?
- Why Can't I Upload Regular Images to Denoise?

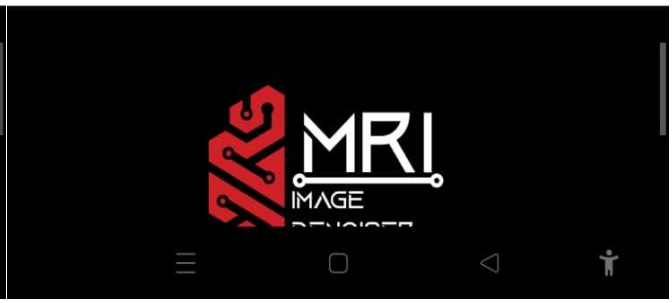
  The Regular Images or Day-to-Day Images Contains Different Noises namely Gaussian Noise, Salt & Pepper Noise, Speckle Noise, Poisson Noise, Impulse Noise and much more but mostly not Ration noise. Whereas our Model is solely to denoise Ration Noise which is more applicable to Medical Images.
- Why am I Unable to Upload Images in PNG/JPG/JPEG/SVG/GIF?
- Why this Algorithm is Working only for Black & White or Grayscale images?

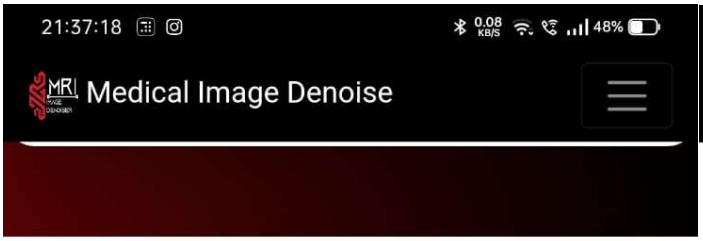
Frequently Asked Questions

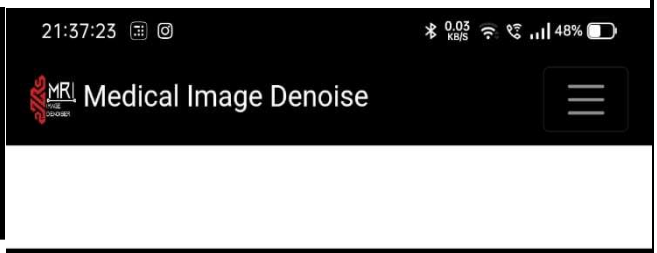
- Is this Algorithm Applicable only for Medical Images?
- Why Can't I Upload Regular Images to Denoise?
- Why am I Unable to Upload Images in PNG/JPG/JPEG/SVG/GIF?

  Our Model is Developed Stricly to Denoise Medical Images & Since Most MRI Images Format is TIFF/TIF and Not PNG/JPG/JPEG/SVG therefore our Model Doesn't Allow User to Upload These Format Images so that it Doesn't Spoil the Uploaded image.
- Why this Algorithm is Working only for Black & White or Grayscale images?









Frequently Asked Questions

- Is this Algorithm Applicable only for Medical Images?
- Why Can't I Upload Regular Images to Denoise?
- Why am I Unable to Upload Images in PNG/JPG/JPEG/SVG/GIF?
- Why this Algorithm is Working only for Black & White or Grayscale images?

  Since MRI Images are Captured in Grayscale Format and the Hard Copy of the Image is in Black and White; It's More Efficient to Output the Model in Grayscale Image Than RGB Values.

MRI IMAGE DENOISER

This Application & Model is created by Presidency University students of batch 2018 - 2022 for our Final Year Project. To know more or to get a clear picture about this work, please access our project report from the link given below. You can also go through the article published about the same & find the link for the code in the Github Repository mentioned below.

Report | Published Article

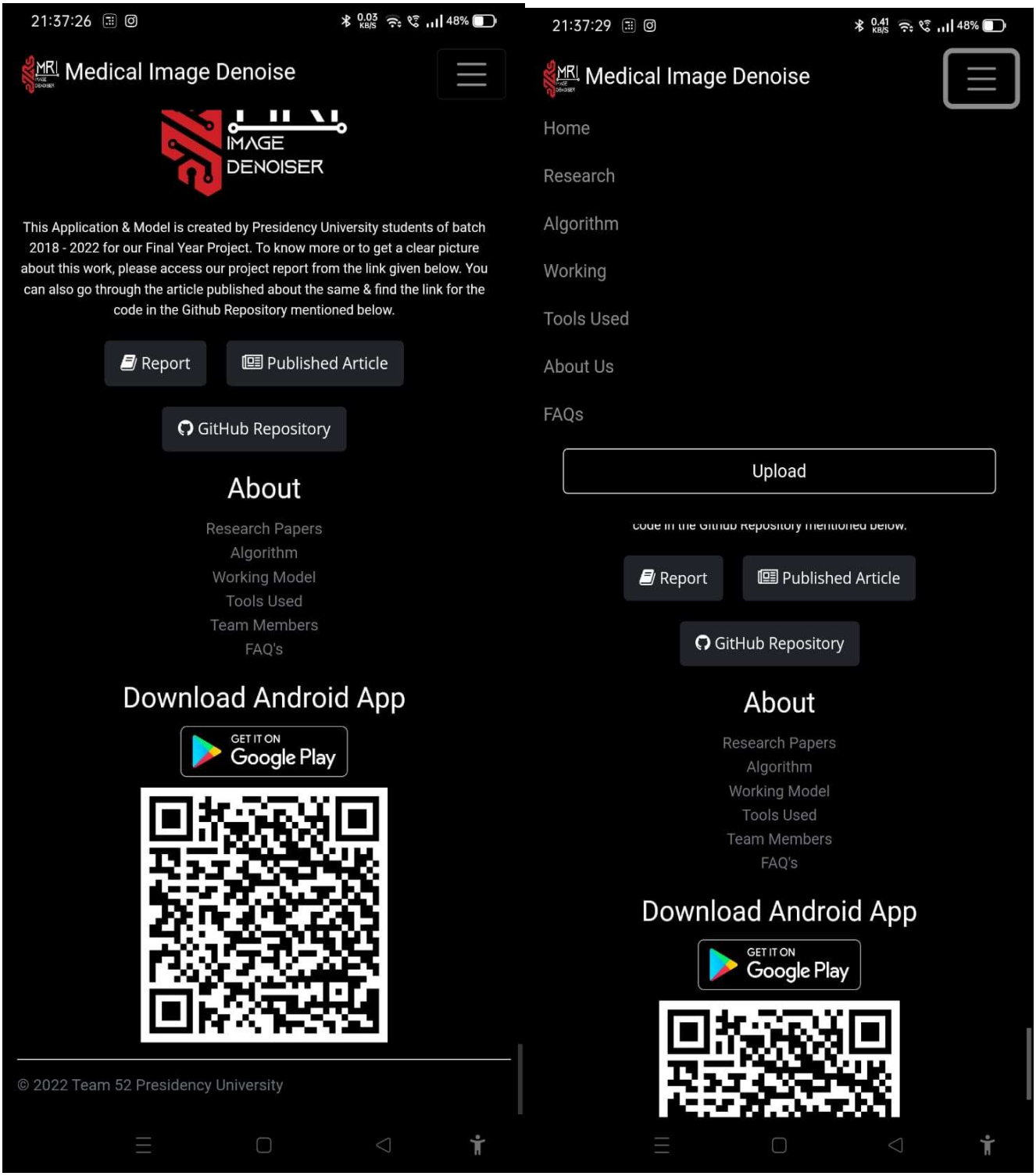
GitHub Repository

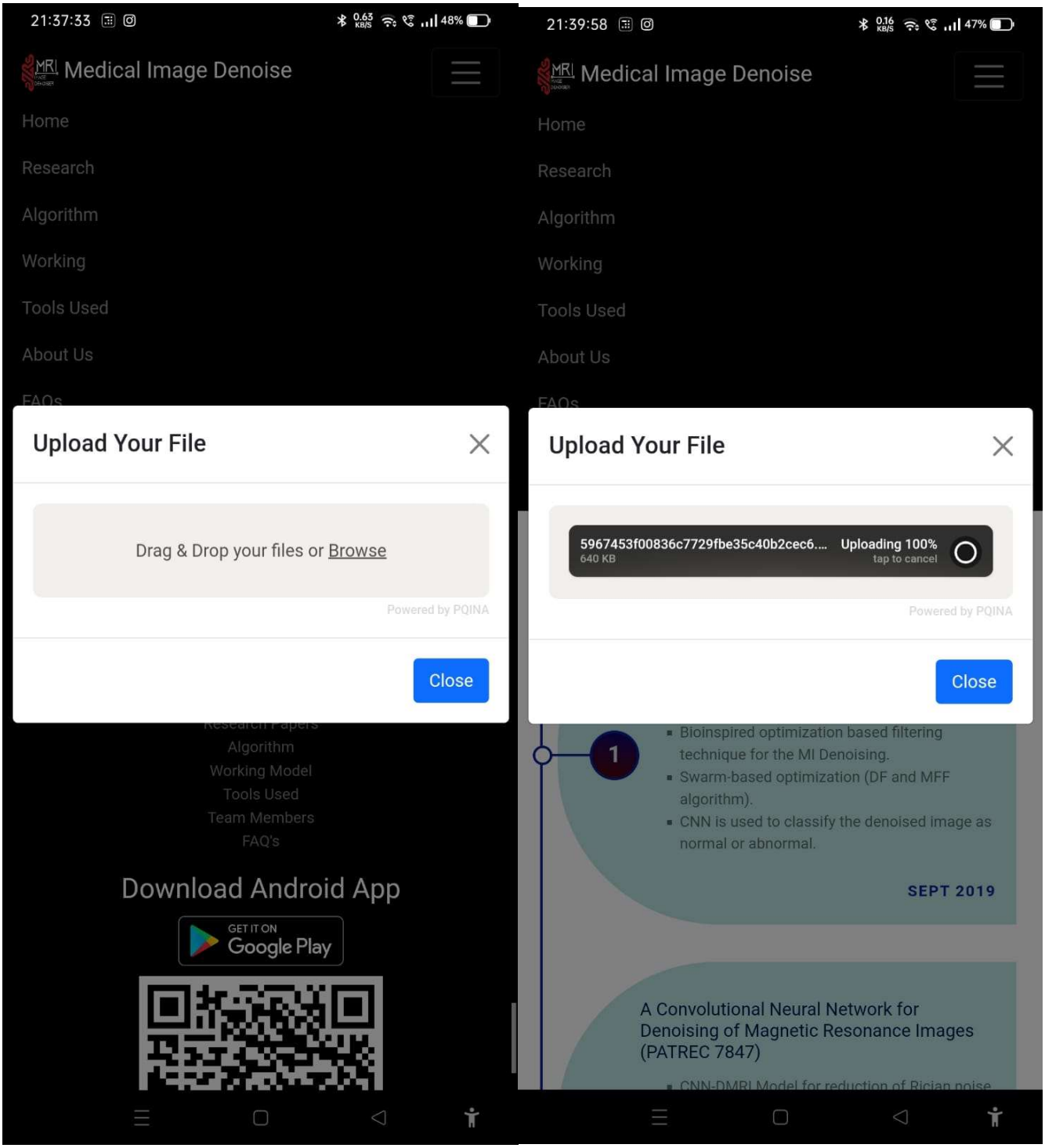
About

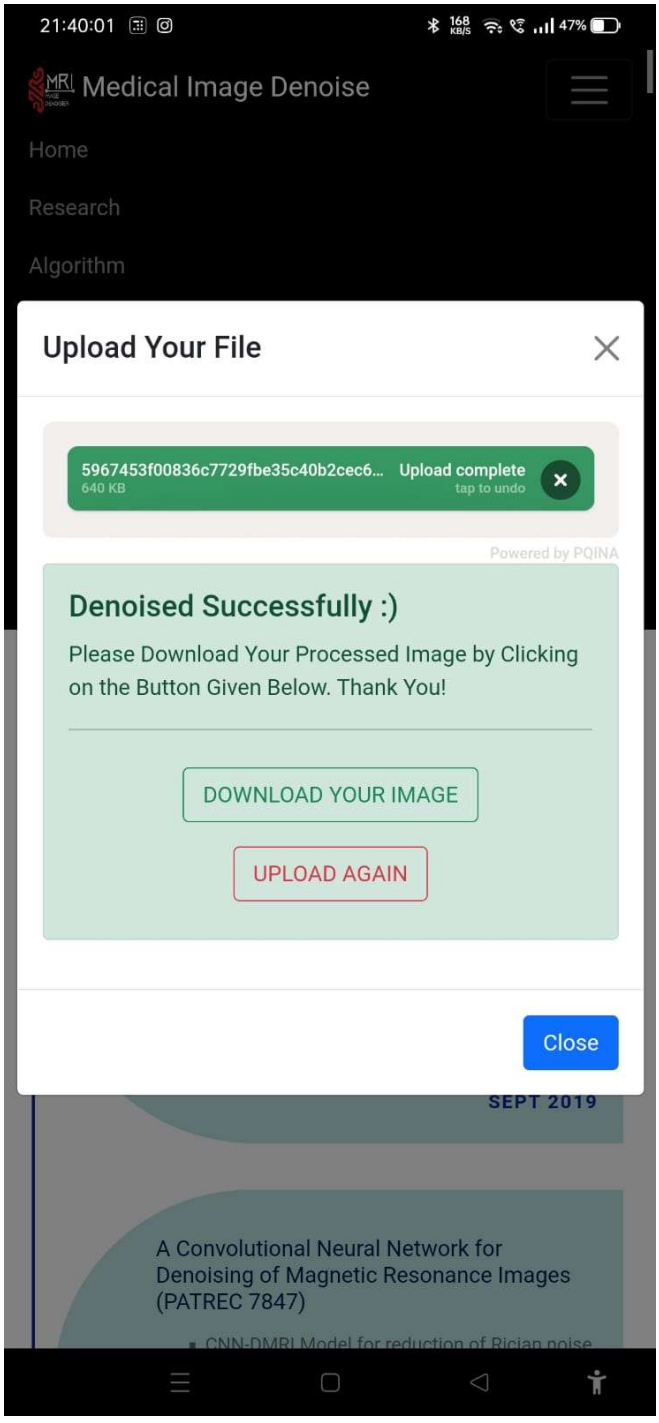
Research Papers
Algorithm
Working Model
Tools Used
Team Members
FAQ's

Download Android App

GET IT ON Google Play


21:37:26
Medical Image Denoise
IMAGE DENOISER
This Application & Model is created by Presidency University students of batch 2018 - 2022 for our Final Year Project. To know more or to get a clear picture about this work, please access our project report from the link given below. You can also go through the article published about the same & find the link for the code in the Github Repository mentioned below.
Report
Published Article
GitHub Repository
About
Research Papers
Algorithm
Working Model
Tools Used
Team Members
FAQ's
Download Android App
GET IT ON Google Play
© 2022 Team 52 Presidency University
21:37:29
Medical Image Denoise
Home
Research
Algorithm
Working
Tools Used
About Us
FAQs
Upload
Report
Published Article
GitHub Repository
About
Research Papers
Algorithm
Working Model
Tools Used
Team Members
FAQ's
Download Android App
GET IT ON Google Play


21:37:33
Medical Image Denoise
Home
Research
Algorithm
Working
Tools Used
About Us
FAQs
Upload Your File
Drag & Drop your files or Browse
Powered by PQINA
Close
Research Papers
Algorithm
Working Model
Tools Used
Team Members
FAQ's
Download Android App
GET IT ON Google Play
21:39:58
Medical Image Denoise
Home
Research
Algorithm
Working
Tools Used
About Us
FAQs
Upload Your File
5967453f00836c7729fbe35c40b2cec6....
640 KB
Uploading 100%
tap to cancel
Powered by PQINA
Close
1
Bioinspired optimization based filtering technique for the MI Denoising.
Swarm-based optimization (DF and MFF algorithm).
CNN is used to classify the denoised image as normal or abnormal.
SEPT 2019
A Convolutional Neural Network for Denoising of Magnetic Resonance Images (PATREC 7847)


21:40:01
Medical Image Denoise
Home
Research
Algorithm
Upload Your File
5967453f00836c7729fbe35c40b2cec6...
640 KB
Upload complete
tap to undo
Powered by PQINA
Denoised Successfully :)
Please Download Your Processed Image by Clicking on the Button Given Below. Thank You!
DOWNLOAD YOUR IMAGE
UPLOAD AGAIN
Close
SEPT 2019
A Convolutional Neural Network for Denoising of Magnetic Resonance Images (PATREC 7847)

9 CONCLUSION & FUTURE WORK

Based on a detailed review, medical picture denoising appears to be a new research topic that has gotten a lot of interest from image and signal processing experts in recent years. As a result, a thorough examination of the major researchers and strategies for medical image denoising is considered. Based on the type of medical image, the studies are divided into three categories: MRI, Ultrasound, and OCT images. The main aspects of the key research that exist in the literature are then evaluated, followed by a simple definition of digital images and medical images and a brief examination of each type of medical images.

Different strategies for denoising medical photographs are thoroughly examined in this research. Ultrasound, MRI, and OCT pictures are all been analyzed. Among other things, the OCT picture is more relevant since it is one of the most prevalent and important modalities used in medical imaging. As a result of its widespread use, getting improved findings for OCT pictures is critical. This article will provide a more solid framework for aspiring researchers looking to identify effective denoising approaches for medical images, particularly OCT images. We anticipate that our review effort will cause many of brainwaves to rise in the future.

10. REFERENCES

1. Ilesanmi, Ademola E., and Taiwo O. Ilesanmi. "Methods for image denoising using convolutional neural network: a review." Complex & Intelligent Systems 7, no. 5 (2021): 2179-2198.
2. Paul, Arati, Ahana Kundu, Nabendu Chaki, Dibyendu Dutta, and C. S. Jha. "Wavelet enabled convolutional autoencoder based deep neural network for hyperspectral image denoising." Multimedia tools and applications 81, no. 2 (2022): 2529-2555.
3. Huo, Yuchi, and Sung-eui Yoon. "A survey on deep learning-based Monte Carlo denoising." Computational Visual Media 7, no. 2 (2021): 169-185.
4. Karkare, Rasika, Randy Paffenroth, and Gunjan Mahindre. "Blind image denoising and inpainting using robust Hadamard autoencoders." arXiv preprint arXiv:2101.10876 (2021).
5. Wang, Feng, Zhiming Xu, Weichuan Ni, Jinhuang Chen, and Zhihong Pan. "An adaptive learning image denoising algorithm based on eigenvalue extraction and the GAN model." Computational Intelligence and Neuroscience 2022 (2022).
6. Elhoseny, Mohamed, and K. Shankar. "Optimal bilateral filter and convolutional neural network based denoising method of medical image measurements." Measurement 143 (2019): 125-135.
7. Feng, Xiangfei, Qinghua Huang, and Xuelong Li. "Ultrasound image despeckling by a hybrid deep network with transferred filtering and structural prior." Neurocomputing 414 (2020): 346-355.
8. Kokil, Priyanka, and S. Sudharson. "Despeckling of clinical ultrasound images using deep residual learning." Computer Methods and Programs in Biomedicine 194 (2020): 105477.
9. Tripathi, Prasun Chandra, and Soumen Bag. "CNN-DMRI: a convolutional neural network for denoising of magnetic resonance images." Pattern Recognition Letters 135 (2020): 57-63.
10. Li, Sanqian, Jinjie Zhou, Dong Liang, and Qiegen Liu. "MRI denoising using progressively distribution-based neural network." Magnetic Resonance Imaging 71 (2020): 55-68.