COSC 4372 “Fundamentals of Medical Imaging”

PAAR DENOISER

AI DENOISER FOR BRAIN TUMOR MRI’S

CONTENT

1. Project report
    - Introduction
    - Method
    - Results and Discussion
    - Conclusion
2. Code
3. Code execution instructions

MEMBERS OF THE TEAM:

Name	PeopleSoft ID	Department	Undergrad/Grad
Pasang Sherpa	-	NSM	Undergrad
Allen Maredia	-	NSM	Undergrad
Ryan Tran	-	NSM	Undergrad
Anand Sabnis	-	NSM	Undergrad

Introduction:

This project aims to implement AI denoising for images. We've used a U-Net architecture that denoises input images and makes them clearer to view and interpret.

Aim 1: Denoising images that are inputted from the user.
Aim 2: Add artificial noise to the inputted image.
Aim 3: Use denoising to try and make the image as clear as possible.
Aim 4: Use the AI model to evaluate metrics to find the aggregate denoising result.

Image denoising with AI is an engaging and impactful project idea that blends technological advancements with real applications that can lead to clearer and more valuable images
Image is the foundation for many real-world applications such as healthcare, automation, satellite imaging, and more.
Enhancing images by removing noise leads to clearer and more interpretable images which help in better analysis and decision making. AI models, especially deep learning models, are very good in terms of image denoising due to their ability to learn complex patterns and representations of data whereas in healthcare and medical imaging such as MRI, and CT scans, reducing noise can lead to clearer diagnostic images and treatment planning.

Methods:

1. Data Preparation:

First, we started with Data Preparation. We had to obtain images to test our U-net framework on, and we wanted images that were visibly different. We then had a set of MRI brain images without noise for training and their corresponding noisy versions. The dataset was imported from Kaggle(.jpeg form) and the images were used to train our model. We didn't remove any images, as we wanted as many as possible. We made an 80-20 testing-training split of the images obtained.

2. Implementing Denoising Networks:

There are a lot of deep learning architectures such as U-Net, GANs, CNN, autoencoders, or other denoising models.

Before we implemented the models, we used image normalization. We decided to go with the U-Net model and Autoencoder models. U-Net is widely known for its effectiveness in image segmentation tasks where it excels at capturing contextual information while preserving spatial details whereas Autoencoder is versatile and adaptable even on small datasets and is based on unsupervised learning.

3. Evaluation Metrics:

Evaluation Metrics are used to help us refine and determine what a good model is. These will show us how close to the actual image our denoised image is. Common metrics for image denoising include PSNR (Peak Signal-to-Noise Ratio), SSIM (Structural Similarity Index), or MSE (Mean Squared Error). We decided to compare all the metrics across both models to have a better grasp of the efficiency of the models.

PSNR stands for "Peak Signal-to-Noise Ratio." It is a metric used to measure the quality of an image or a video signal. It quantifies the difference between the original image and the compressed or processed image by comparing the maximum possible power of a signal to the power of corrupting noise.

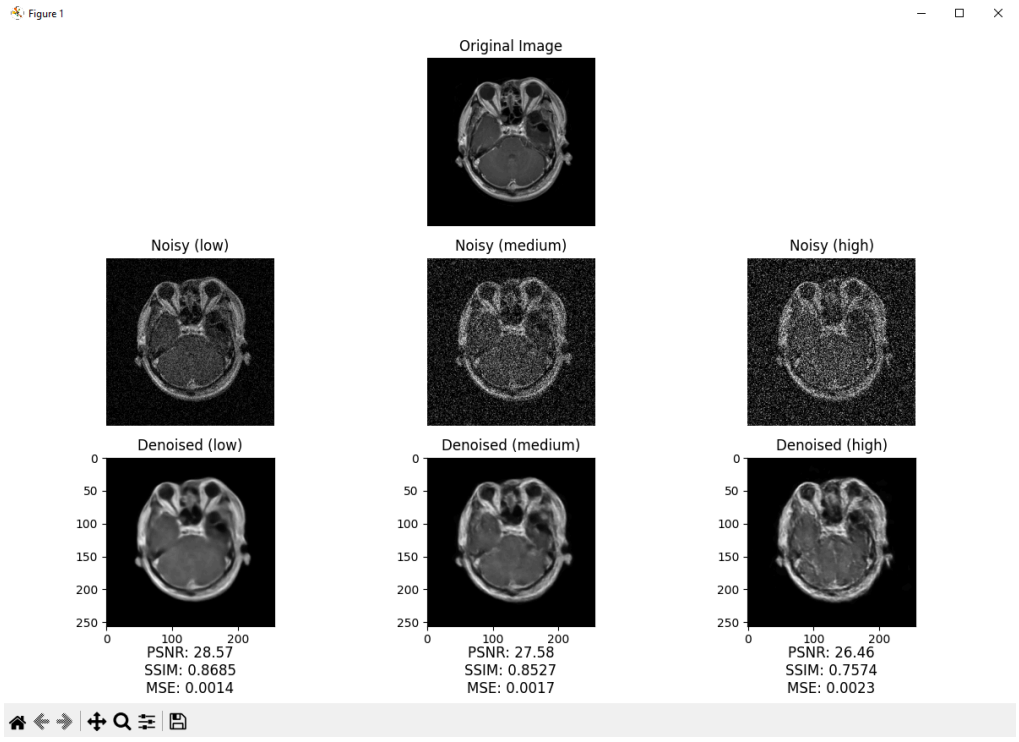
SSIM stands for "Structural Similarity Index Measure." It is another metric used to evaluate the similarity between two images. SSIM measures the perceived change in structural information, luminance, and contrast of images. Higher SSIM values generally indicate greater similarity between the images being compared.

MSE stands for "Mean Squared Error." It is a common measure used in image processing or signal processing to quantify the average squared differences between corresponding pixels or values in two images. Lower MSE values indicate less difference between the images being compared, suggesting higher similarity or better quality.

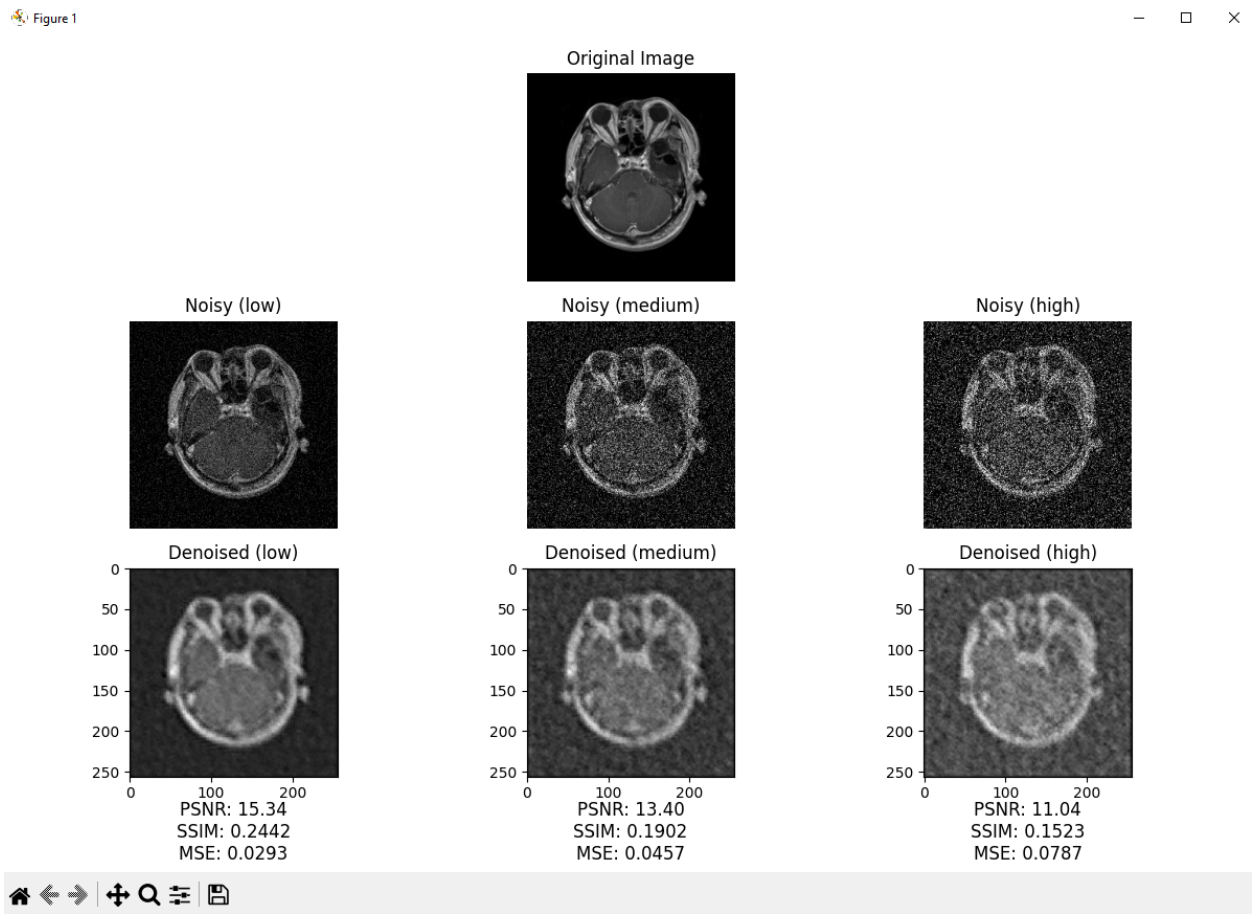
4. Training and Analysis:

We trained the U-net model using the images in the brain tumor MRI dataset. This dataset contains 7023 images of human brain MRI images which are classified into 4 classes: glioma - meningioma - no tumor and pituitary. We picked the glioma subset to train our model. The glioma subset contains 1321 images that all show a glioma tumor in the brain.

The training and testing data split was 80-20(80% of images were used for training, whereas 20% were used for testing). This allowed us to extensively train and test our model with different looking images.



U-Net Architecture

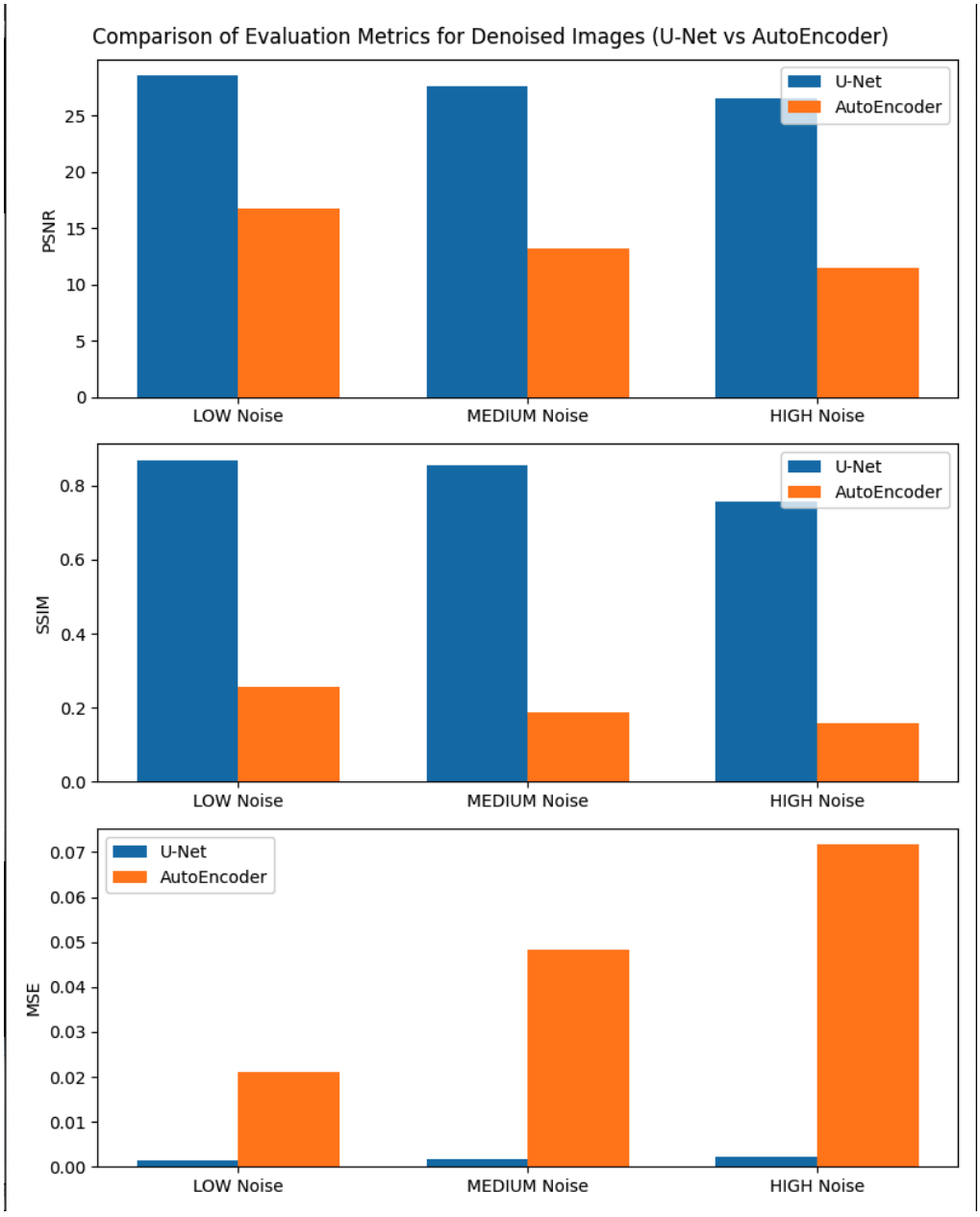


AutoEncoder

Results and Discussion:

The choice between U-Net and autoencoder for image denoising may depend on various factors such as the nature of noise in the images, the level of detail preservation required, computational resources, and the specific requirements of the denoising task. Experimentation and evaluation using both architectures on a particular dataset or problem can help determine which model performs better for a given scenario.

The main results are shown in the bar graph below:



Conclusions

From the results, we can see on average, the U-Net architecture performed roughly twice as well when comparing the models with the PSNR evaluation metric and performed roughly three times as well when comparing with the SSIM evaluation metric. Additionally, the MSE increased linearly for the model using AutoEncoder as the level of noise increased while the MSE remained fairly constant for the model using U-Net despite changes in noise level.

In summary, the U-Net model consistently produced higher-quality denoised images when presented with brain tumor MRIs."

DATASET USED:

https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset/code

Code Execution Instructions

Packages needed:
-Numpy
-Matplotlib
-Scikitlearn
-Tensorflow
-PIL
-Tkinter

Execution Instructions:
-Download the two models (U-Net, AutoEncoder) (optional if you wish to train the model yourself)
https://drive.google.com/file/d/1-PtuZHfryAqXFHLLkWcjyhny-F0iY5me/view?usp=drive_link
https://drive.google.com/file/d/1-PtuZHfryAqXFHLLkWcjyhny-F0iY5me/view?usp=drive_link

-Download the files to train the models (UNetTrain.py, AutoEncoderTrain.py)

-Download the dataset
https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset/code

-Configure the path of the training set within the two training files and run the file to create the two models

-Configure the path of created (or downloaded) models inside the AIDenoisingUNET.py file along with the AIDenoisingAutoEncoder.py file

-Run the file, select an MRI image from the testing set, and view the results