Image Denoising using Bayesian Methods

Sharvi Tomar
stomar2

Aakansha Singh
singh107

Neha Bharadwaj
nehab3

December 12, 2022

1 Problem Statement

Option 2 - The aim of this project is to review, summarize, and illustrate a special topic, Image Denoising using Bayesian Methods, as discussed in modern Bayesian literature. We will use a simulated dataset (described in the Data description) and implement Bayesian techniques to denoise and restore the true images with as little information loss as possible.
In our work, we have taken all the suggestions into account and tried to address the questions posed by Professor in the proposal review. In the following sections, we will explain the basic idea about Image denoising and implement a published methodology to denoise images from the simulated dataset.

2 Introduction

A major component in the process of image processing is the denoising of images which essentially is the process of removing noise from an image to restore the true image in order to minimize information loss from the image due to noise. Image denoising plays an important role in various tasks such as image classification, image restoration, and image segmentation. So far researchers have proposed many different methods to perform image denoising, we will be focusing on discussing the Bayesian method for the process in detail.

2.1 Noise and its types

Noise is a random variation in color or brightness information. The cause of noise in an image is generally unsuitable environment(low light, dust particles, etc.,) or the device/ sensor used to collect the image. Basically noise can be understood as corrupted pixels in the image. Furthermore, the noise can also be due to an interference in the transmission channel.
There are different kinds of noises based on pattern and different probabilistic properties. Few important ones are discussed below-

Gaussian Noise - It is a statistical noise having a probability density function of Normal distribution and is evenly spread throughout the image signal. When this noise is added into an image the original pixel value of the image changes to original pixel plus a random

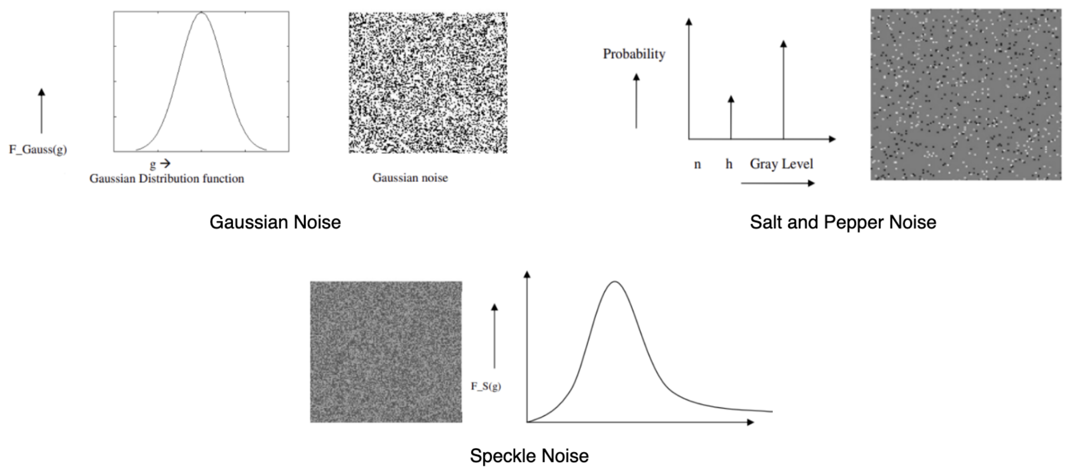

Figure 1: Types of Noise

Gaussian noise value. Figure 1 shows the Gaussian noise and it distribution graph.(1)
Salt and Pepper Noise - This noise generates white and black pixels at random intervals and it is caused when there is error in transfer of data. This noise can be removed using median filter. (1)
Speckle Noise - It is a multiplicative noise and it reduces the quality of image by scattered waves. This makes difficult to see any fine details in the image. (1)

2.2 Image denoising Techniques

There are various techniques used to denoise an image such as Spatial filtering and Transform domain filtering. In contrast to most techniques that take into consideration the maximum likelihood or the best estimate for data analysis, the Bayesian approach helps us to take into account a complete model and also account for the uncertainties in the model and parameter values. The basic idea is to define a parametric prior and adjust these parameters for each image that is to be denoised in a way that is not too computationally intensive and is able to yield accurate results[12].

2.3 Using Bayes theorem in denoising

The problem of image denoising in Bayesian terms be modeled as y= x + n, where n represents additive Gaussian noise with a certain known standard deviation[5].In Bayesian image processing, the objective image can be inferred based on the posterior probability distribution that is the distribution of y given x [8]. An important bayesian framework for image processing is based on Markov random fields (MRFs) which is an undirected graph of the probability distribution of any pixel depending on the values of neighboring pixels. Going forward we will in-depth discuss the prior information, data description, and the implementation of performing denoising using Markov random fields and Gibbs sampling.

2.4 Benefits of Denoising

Owing to the influence of environment, transmission channel, and other factors, images are inevitably contaminated by noise during acquisition, compression, and transmission, leading to distortion and loss of image information[5]. In our review, we will illustrate various examples from different domains where the need for image denoising is indispensable such as Medical Imagery (CT Scans, X-rays, etc.,). While denoising an image might not affect the brightness or contrast of the image directly, removal of the unwanted noise helps make the image clearer and hence appear slightly brighter. This helps people in fields like Architecture, medicine, photography, etc. to extract useful information from these denoised images.

3 Literature Review

As discussed in [5], the methods adopted for the task of image denoising can be broadly grouped into three categories: spatial domain methods, transform domain methods, and CNN-based methods. The spatial domain methods are classical methods that address noise removal by computing the correlation of pixel values (or image patches) in the original image. These methods can be further categorized into spatial domain filtering and variational domain methods. Spatial domain filtering methods use low pass filtering on pixel groups and the variational domain methods involve using image priors and energy functions for obtaining denoised images. We have adopted a variational domain approach for our implementation and Section 5 covers details of our methodology.
A very popular algorithm in the field of image processing is the non-local means (NL-means) algorithm for image denoising [1]. The algorithm is based on local averaging of all pixels (mostly, in a certain window size for computational feasibility similarity). A weighted averaging flavor of local averaging is used in which the weights for a pixel represent the similarity information of that pixel with the target pixel. The NL-means algorithm has gained popularity because of its effectiveness in retaining image detail and clarity. The NL means not only compares the grey level in a single point but the geometrical configuration in a whole neighborhood. This fact allows a more robust comparison than neighborhood filters. Figure 1 illustrates the fact, that pixel q3 has the same grey level value as pixel p, but the neighborhoods are much different and therefore the weight w(p, q3) is nearly zero.
One of the major concerns while performing Bayesian analysis is the selection of an appropriate prior in order to not adversely influence the posterior distribution. In the context of Bayesian clinical trial design, well-chosen priors are important to ensure that posterior-based decision rules have good Frequentist properties[10].
A prior effective sample size (ESS) can help us in understanding the prior assumptions and guide the calibration of prior variance and hyperparameter parameters. The ESS can be defined as the interpolated value that minimizes the distance between the prior and the posterior distribution. For instance, a prior ESS that is almost equivalent to our actual data sample size can mean that the prior is over informative and one with an extremely low value can hint towards a non-informative prior. The ESS is desired to be a value that is small enough so

that the initial decisions are dominated by the data instead of the prior[10].

Traditional denoising algorithms assume that the image is smooth everywhere and thus use smooth image models to process them. A smooth MRF is generally used for image-denoising applications and assumes that neighboring pixels have similar colors. Also, it is very difficult to optimize the model due to the complex dependency of the neighboring variables. On the other hand, we cannot forget that images also contain some textures besides smooth regions, and using the smooth MRF models can reduce the effectiveness of the results and lose image features/ information. In order to preserve image textures, a Hierarchical MRF image model is employed that assumes that an image is made up of disjoint texture regions. The Hierarchical MRF model works on three layers - image texture (region labels), noise-free image color, and noisy image color. As per the denoising algorithm, the noise-free image has l regions and each region has its own parameters, calculates local variance, and finds clusters with kmeans, the goal is to estimate the MRF parameters for each iteration. This method can express both smooth and texture signals such that image texture is preserved while suppressing noise systematically.[2]

Another possible method of image denoising is to consider a case where several noisy images are available in order to generate the original image and a Bayesian approach is adopted in order to create the posterior probability distribution of the denoised image. The performance of the specified method can be analyzed using multiple images. The Bayesian image processing discussed is based on MRF(Markov random fields) which is an important framework in the field of processing[6].

The experiment was focused on implementing a discrete MRF prior combined with the Bayesian approach. One of the disadvantages of using a discrete MRF prior is computational complexity. Hence in order to propose a possible method for using multiple images instead of a single image to denoise and get an original image would be to use a discrete MRF prior combined with a Fast Fourier transform. The idea is to use K noisy images to find an image x-hat that maximizes the posterior probability distribution and this would be our resulting denoised image. In order to implement the Bayesian perspective, a parameterized belief propagation method is used and the parameters of the model are estimated using the expectation maximization algorithm[6].

The state-of-the-art approach for image denoising involves deep learning techniques (deep prior) along with Bayesian inference. The image output is generated using convolutional neural networks with random inputs. Then the posterior inference is calculated using a stochastic gradient descent approach. Bayesian inference helps to avoid overfitting by adding suitable priors over the parameters and thereafter uses posterior distributions to calculate the uncertainty. Moreover, advancements are concurrently going to improve the image denoising process using Deep learning and Bayesian techniques [3].

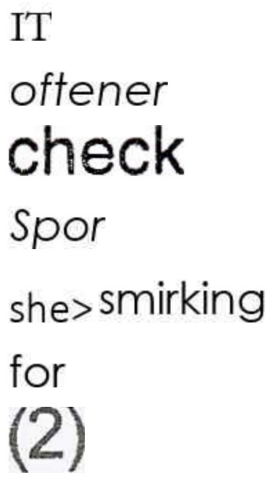

Figure 2: Snippet of Word MNIST Data

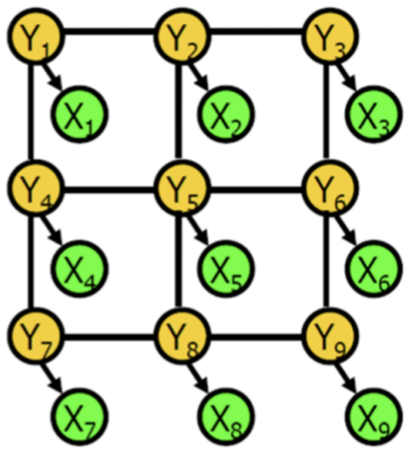

Figure 3: Pairwise Markov Random Field [13]

4 Methodology

To decrease image noise while minimizing loss of original features as well as improving the signal-to-noise ratio(SNR). The problem can be modeled as follows:

$$x = y + \eta$$

where x is the observed noisy image, y is the clean image and η is the additive white Gaussian noise(AWGN).

We have adopted a variation denoising method which is a classical approach to image-denoising from this reference [13].

4.1 Dataset [11]

We initially proposed to present toy examples for image denoising using the Medical MNSIT dataset. However, after initial processing, we encountered computational challenges due to the high resolution of the images. Then we chose low-resolution images to implement the denoising algorithm.
The dataset consists of images of short English words(2). The dataset mainly comprises black-and-white images of textual data. These images are manually cropped from naturally scanned documents. The data set we have used is a subset of the main words MNIST dataset, where the original had 1.2M images, however, the subset we have used comprises of 10,000 images from the main dataset.

4.2 Implementation

In order to denoise our input image that existingly has gaussian noise added on top of it, we have implemented a function that gives estimated posterior probabilities with a threshold of

0.5.

Image Prior Used

It is assumed that the image has sparse gradients so we have used sparse gradients prior which penalizes for large gradients.

$$R(y_{i,j}|X) = \sum_{z \in N_{y_{(ij)}}} ||\frac{z - y_{ij}}{z}||$$

4.2.1 Applying gibbs sampling based on Markov Random Field

Markov Random Field is a graph model of a joint probability density with each node being a random variable. In this undirected graph, each edge defines the local influences among pairs of nodes. So each pixel in the image has a value that denotes its color. We consider four neighboring nodes adjacent to a pixel having similar colors. This represents the smoothness of an image. Except for a few abruptness such as edges in an image, real-world images are more or less smooth. Analogous to the Bayesian network, the Markov Random Field network provides the conditional independence relationship between nodes but MRF is undirected and acyclic, unlike Bayesian network(3).

Gibbs sampling

Gibbs sampling is a Markov Chain Monte Carlo algorithm for sampling distributions in the multivariate case where it is difficult to sample directly and the joint distribution is unknown. For multi-dimensional cases (for example a 2D or 3D image), Gibbs sampling generates a chain of Markov chain samples, that are related to their nearby samples. We discard the samples from the initial steps of the chain as they may not accurately present the required distribution.
In order to implement a sample experiment we have used 1000 iterations for each image combined with 100 burn in iterations.

4.2.2 Calculating Energy function and Loss

The energy value of a pixel in the observed noisy image is a sum of three components: loss, sparse gradient prior, and window-wise products. The computation of energy depends on MRF.

$E_{y_{ij}}$ = Loss(L) + Sparse Gradient Prior(R) + Window-wise Products(W)

$$E(y_{ij}|X) = E(y_{ij}|N_{y_{ij}})$$

$$E(y_{ij}|X) = L(y_{ij}|N_{y_{ij}}) + \lambda_r R(y_{ij}|N_{y_{ij}}) - \lambda_w W(y_{ij}|N_{y_{ij}})$$

$$p(y_{ij}|N_{y_{ij}}) = \frac{1}{z} exp(-E(y_{ij}|X))$$

where Z is a normalizing constant.

From the above equation, we can imply that the higher the energy lower is its chance of getting sampled.

$$\hat{x} \in \arg\min_x E(x)$$

$$\hat{x} = \arg\max_x P(x|y)$$

$$\hat{x} = \arg\max_x \frac{P(y|x)P(x)}{P(y)}$$

$$\hat{x} = \arg\max_x \log P(y|x) + \log P(x)$$

The loss is representative of the difference (or distance) between the neighboring four pixels and the pixel in the noisy image as well as the difference with the corresponding pixel in the denoised image.

$$L(y_{i,j}|X) = \sum_{z \in N_{y_{(ij)}}} ||z - y_{ij}|| + \lambda ||x_{ij} - y_{ij}||$$

Due to the limitation of the L-1 and L-2 norm, a robust method to outliers called the Lorentzian function is used.

$$\rho(z, \sigma) = log(1 + \frac{1}{2}(\frac{z}{\sigma})^2)$$

where σ is the bandwidth.

$$L(y_{i,j}|X) = \sum_{z \in N_{y_{(ij)}}} \rho(z - y_{ij}, \sigma) + \rho(x_{ij} - y_{ij}, \sigma)$$

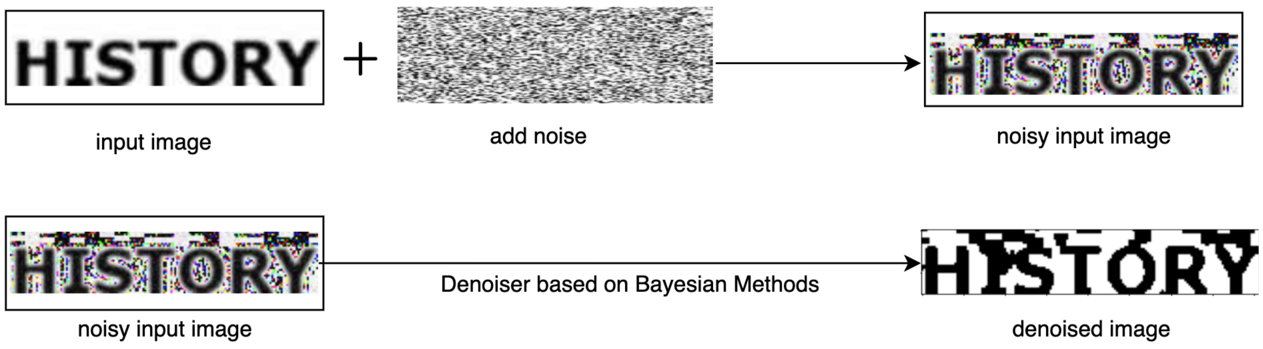

Figure 4: Denoising Process

4.3 Result Analysis

To analyze the performance of the image-denoising method, we inspected the outputs visually. Based on the image attributes we applied a gaussian noise with a standard deviation of 20 and clipped out the boundary values before denoising the images and fetching a posterior probability for the true estimates.The variation in the noise distribution can significantly cause variations in output as we increased the noise the outputs were not as clear for the same number of iterations and burnin. For visual inspection, three criteria were considered: (1) retention of edges, (2) preservation of edges, and (3) degree of artifacts. The produced results seem to perform well on all three criteria which proves the efficacy of variational denoising methods of image denoising. The energy convergence results are also pretty interesting. From the Energy log graph, we find that although image Y was randomly initialized, the energy function converged very fast to a lower value.

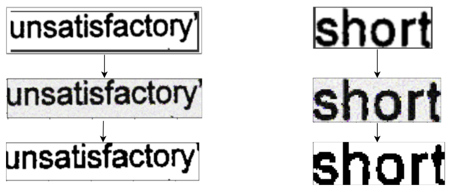

Figure 5: Sample Results

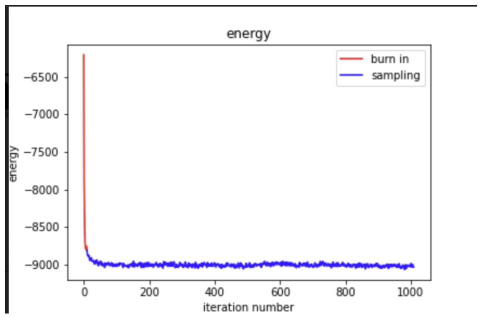

Figure 6: Energy log graph

5 Conclusion

This project helped us to understand the idea behind Image processing and Image denoising. We also learnt how the use of Bayesian techniques can enhance the performance of an image denoiser algorithm. Though Image denoising has a lot of applications in real world, there are still some challenges that linger around it.

The mathematical hurdle in solving the image-denoising problem is an ill-posed inverse problem. Often times a significant amount of information gets lost from the image like (fine details in medical imagery) after denoising. Ideally, the process of denoising should preserve the information present in the original image along with texture and edges however it's not always easy to do that. The irregular edges of the image, rough texture and additional information irrelevant/ extremely high noise to the original image are also some challenges that one can face.

Apart from all these challenges, Image denoising remains to be very active field of research and development. Variety of optimization techniques are being incorporated to enhance the result of existing algorithms for image denoising. Advancements in Deep Learning and Bayesian inference methodologies are paving the path for new generation Image processing systems.

References

[1] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65 vol. 2, 2005.

[2] Yang Cao, Yupin Luo, and Shiyuan Yang. Image denoising based on hierarchical markov random field. *Pattern recognition letters*, 32(2):368–374, 2011.

[3] Zezhou Cheng, Matheus Gadelha, Subhransu Maji, and Daniel Sheldon. A bayesian perspective on the deep image prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5443–5451, 2019.

[4] Nicolas Dobigeon, Alfred O Hero, and Jean-Yves Tourneret. Bayesian sparse image reconstruction for mrfm. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2933–2936. IEEE, 2009.

[5] Linwei Fan, Fan Zhang, Hui Fan, and Caiming Zhang. Brief review of image denoising techniquesc. *Visual Computing for Industry, Biomedicine, and Art*, 2(1):1–12, 2019.

[6] Shun Kataoka and Muneki Yasuda. Bayesian image denoising with multiple noisy images. *The Review of Socionetwork Strategies*, 13(2):267–280, 2019.

[7] Manpreet Kaur and Sunny Behal. Study of image denoising and its techniques. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3, 02 2013.

[8] H Shawn Kim, Cheolkon Jung, Sunghyun Choi, Sangseop Lee, and Joong Kyu Kim. A novel approach for bayesian image denoising using a sgli prior. In *Pacific-Rim Conference on Multimedia*, pages 988–993. Springer, 2009.

[9] Savvas Melidonis, Paul Dobson, Yoann Altmann, Marcelo Pereyra, and Konstantinos C Zygalakis. Efficient bayesian computation for low-photon imaging problems. *arXiv preprint arXiv:2206.05350*, 2022.

[10] Satoshi Morita, Peter F Thall, and Peter Müller. Evaluating the impact of prior assumptions in bayesian biostatistics. *Statistics in biosciences*, 2(1):1–17, 2010.

[11] Tushar Pawar. words mnist classification. https://www.kaggle.com/datasets/backalla/words-mnist, 2017.

[12] Umesh Rajashekar and Eero P Simoncelli. Multiscale denoising of photographic images. In *The Essential Guide to Image Processing*, pages 241–261. Elsevier, 2009.

[13] Chang Yue. Markov random fields and gibbs sampling for image denoising. *Electrical Engineering Stanford University*, 2021.

Appendix

Individual contributions from each team member are as follows -

1. Aakansha Singh
    - Introduction (Image denoising Overview)
    - Literature review - reviewed and summarised("Image denoising based on hierarchical Markov random field").[2]
    - Literature review - reviewed and summarised("A Bayesian Perspective on the Deep Image Prior ").[3]
    - Code Implementation and debugging
    - Methodology and Implementation
    - Analysis, Result and Conclusion
    - Report preparation and Editing
2. Neha Bharadwaj
    - Introduction (Image denoising Overview)
    - Literature review - reviewed and summarised("Bayesian image denoising with multiple noisy").[6]
    - Literature review - reviewed and summarised("Evaluating the impact of prior assumptions in bayesian biostatistics").[10]
    - Code Implementation and debugging
    - Methodology and Implementation
    - Analysis and Result
    - Report preparation and Editing
3. Sharvi Tomar
    - Literature review - reviewed and summarised("Brief review of image denoising techniques").[5]
    - Literature review - reviewed and summarised("A non-local algorithm for image denoising").[1]
    - Code Implementation
    - Methodology and Implementation
    - Analysis and Result

```
from google.colab import drive
drive.mount('/content/drive')

     Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import os
os.chdir("/content/drive/MyDrive/Colab Notebooks/STAT 431")
```

```
import math
import numpy as np
from copy import deepcopy
import matplotlib.pyplot as plt
from scipy import ndimage, misc
from PIL import Image
```

```
def add_gaussian_noise(img, sigma=5):
    clone = deepcopy(img)
    noise = np.abs(np.random.normal(0, scale=sigma, size=img.shape))

    subtraction = np.asarray(clone - noise, dtype=np.uint8)
    addition = np.asarray(clone + noise, dtype=np.uint8)

    boolz = np.array(clone == 255)
    gaussian_noise = np.where(boolz, subtraction, addition)

    plt.imshow(gaussian_noise, cmap=plt.get_cmap('gray'))
    plt.show()
    return gaussian_noise
```

```
import imageio

sigma = 20
for x in range(0,5):
  orig_img = imageio.imread("img"+str(x+1)+".png")
  orig_img.shape
  noisy_img = add_gaussian_noise(orig_img, sigma=sigma)
  imageio.imwrite("noisy_img_"+str(x)+".png", noisy_img)
```

```
# Function denoise_image() is the entry function. It calls another function get_posterior to get the estimated posterior probabilities
# p(Y=1|Y_neighbor). Setting the threshold to 0.5, we can get the restored image array Y from the posterior.
# Next, we strip the edges of the image array and return it. The following load_image function explains why we need to strip the array.

def denoise_image(filename, burn_in_steps, total_samples, logfile):
    posterior = get_posterior(filename, burn_in_steps, total_samples, logfile=logfile)
    denoised = np.zeros(posterior.shape, dtype=np.float64)
    denoised[posterior > 0.5] = 1
    return denoised[1:-1, 1:-1]
```

```
# In get_image(), we first read the PNG image into a numpy array, then we convert the RGB image to grayscale and re-scale the pixels to {-1,
# Here we add 0 paddings to the edges to help take care of corner cases when we search for each pixel’s neighbor later.
def get_image(filename):
    my_img = plt.imread(filename)
    img_gray = np.dot(my_img[..., :3], [0.2989, 0.5870, 0.1140])
    img_gray = np.where(img_gray > 0.5, 1, -1)
    img_padded = np.zeros([img_gray.shape[0] + 2, img_gray.shape[1] + 2])
    img_padded[1:-1, 1:-1] = img_gray
    return img_gray
```

```
for i in range(0,5):
  X = get_image("/content/drive/MyDrive/Colab Notebooks/STAT 431/noisy_img_"+str(i)+".png")
  plt.imshow(X)
  plt.show()
```

```
def sample(i, j, Y, X):
    markov_blanket = [Y[i - 1, j], Y[i, j - 1], Y[i, j + 1], Y[i + 1, j], X[i, j]]
    w = ITA * markov_blanket[-1] + BETA * sum(markov_blanket[:4])
    prob = 1 / (1 + math.exp(-2*w))
    return (np.random.rand() < prob) * 2 - 1
```

```
def get_posterior(filename, burn_in_steps, total_samples, logfile):
    X = get_image(filename)
    posterior = np.zeros(X.shape)
    print(X.shape)
    Y = np.random.choice([1, -1], size=X.shape)
    energy_list = list()
    for step in range(burn_in_steps + total_samples):
        for i in range(1, Y.shape[0]-1):
            for j in range(1, Y.shape[1]-1):
                y = sample(i, j, Y, X)
                Y[i, j] = y
                if y == 1 and step >= burn_in_steps:
                    posterior[i, j] += 1
        energy = -np.sum(np.multiply(Y, X))*ITA-(np.sum(np.multiply(Y[:-1], Y[1:]))+np.sum(np.multiply(Y[:, :-1], Y[:, 1:])))*BETA
        if step < burn_in_steps:
            energy_list.append(str(step) + "\t" + str(energy) + "\tB")
        else:
            energy_list.append(str(step) + "\t" + str(energy) + "\tS")
    posterior = posterior / total_samples

    file = open(logfile, 'w')
    for element in energy_list:
        file.writelines(element)
        file.write('\n')
    file.close()
    return posterior


def save_energy_plot(filename):
    x = np.genfromtxt(filename, dtype=None, encoding='utf8')
    its, energies, phases = zip(*x)
    its = np.asarray(its)
    energies = np.asarray(energies)
    phases = np.asarray(phases)
    burn_mask = (phases == 'B')
    samp_mask = (phases == 'S')
    assert np.sum(burn_mask) + np.sum(samp_mask) == len(x), 'Found bad phase'
    its_burn, energies_burn = its[burn_mask], energies[burn_mask]
    its_samp, energies_samp = its[samp_mask], energies[samp_mask]
    p1, = plt.plot(its_burn, energies_burn, 'r')
    p2, = plt.plot(its_samp, energies_samp, 'b')
    plt.title("energy")
    plt.xlabel('iteration number')
    plt.ylabel('energy')
    plt.legend([p1, p2], ['burn in', 'sampling'])
    plt.savefig('%s.png' % filename)
    plt.close()


def save_denoised_image(denoised_image , i):
    plt.imshow(denoised_image , cmap = 'gray')
    plt.title("denoised image")
    plt.savefig('Output/denoise_img_'+ str(i)+".png")
    plt.close()


if __name__ == '__main__':
  ITA = 1
  BETA = 1
  total_samples = 1000
  burn_in_steps = 10
  logfile = "Output/log_energy"
  for i in range(0,5):
    denoised_img = denoise_image("noisy_img_"+str(i)+".png", burn_in_steps=burn_in_steps,
                                 total_samples=total_samples, logfile=logfile)
    save_energy_plot(logfile)
    save_denoised_image(denoised_img , i)
```

✓ 1m 5s completed at 4:25 PM