CS5340 : Uncertainty Modeling in AI

Final Project

Shrey Jain (A0178310L)
Manasa Kashyap (A0178462W)

Introduction

This report describes our implementation of the following:

1) Image Denoising
    a) Using Gibbs Sampling
    b) Using Variational Inference
2) Image Segmentation using EM algorithm

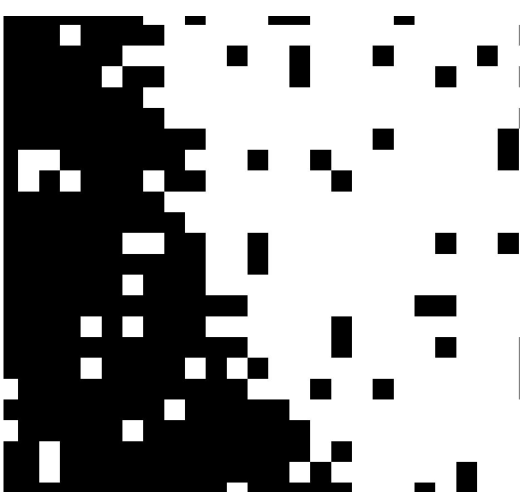
Image Denoising

Given a binary image (ie black or white pixels), the objective is to partially reconstruct the original image as close as possible. To accomplish this we use 2 approaches. a) Gibbs Sampling and b) Variational Inference. We assume that the reader is familiar with the 2 approaches above as well as the Ising model. In our report, we shall only describe the salient features of our implementation and our choice of functions/parameters used.

How to choose parameters J, W, c, sigma?

The choice of parameters J (coupling strength) and sigma (variance of Gaussian observation model) are very critical for determining the results of the denoising using Gibbs sampling. Similarly the parameters W and c serve the same purpose in determining the results of denoising using Variational Inference.

In the raw (noisy) images we notice that - if a pixel is surrounded by 3 or more pixels of the opposite color, it is likely to originally be the opposite color. Refer to the image below:



This implies that we can establish the following **principles** while selecting the parameters:

1) If a pixel is surrounded by 3 or 4 pixels of opposing color, the probability of ‘flipping’ the pixel should be very high.
2) If the pixel is surrounded by 0, 1 or 2 pixels of opposing color, the probability of ‘flipping’ the pixel should be very low.

Notice that the value of sum_neighbours $\sum_{s \subset nbr(t)} x_s x_t$ can only be in [4, 2, 0, -2, -4]. If there are 4 pixels of opposing color, the value of the summation will be -4, if there are 3 pixels of opposing color, the value of the summation will be -2, and if there are 2 pixels of opposing color, then the value will be zero. Therefore, if we can calculate the probability of flipping as a function of sum_neighbours in [0, -2, -4], we can find the best parameters (J, sigma) to use.

In the following table we summarise the values of $pflip_0$ $pflip_{-2}$ $pflip_{-4}$ for each value of sigma in range [0.1, 0.6] and J in [1, 3]. Intuitively, sigma is the weakness of the local evidence while J is the strength of the ‘neighbouring effects’. A higher sigma makes the local evidence weaker and more prone to flipping, while a higher J makes the strength of the coupling function greater and making the pixel more likely to flip when surrounded by opposing neighbours.

	J = 1	J = 2	J = 3
Sigma = 0.1	$pflip_{-4}$ = ~0.00 $pflip_{-2}$ = ~0.00 $pflip_0$ = ~0.00	$pflip_{-4}$ = ~0.00 $pflip_{-2}$ = ~0.00 $pflip_0$ = ~0.00	$pflip_{-4}$ = ~0.00 $pflip_{-2}$ = ~0.00 $pflip_0$ = ~0.00
Sigma = 0.2	$pflip_{-4}$ = ~0.00 $pflip_{-2}$ = ~0.00 $pflip_0$ = ~0.00	$pflip_{-4}$ = ~0.00 $pflip_{-2}$ = ~0.00 $pflip_0$ = ~0.00	$pflip_{-4}$ = ~0.00 $pflip_{-2}$ = ~0.00 $pflip_0$ = ~0.00
Sigma = 0.3	$pflip_{-4}$ = ~0.00 $pflip_{-2}$ = ~0.00 $pflip_0$ = ~0.00	$pflip_{-4}$ = ~0.00 $pflip_{-2}$ = ~0.00 $pflip_0$ = ~0.00	$pflip_{-4}$ = 0.86 $pflip_{-2}$ = ~0.00 $pflip_0$ = ~0.00
Sigma = 0.4	$pflip_{-4}$ = 0.01 $pflip_{-2}$ = ~0.00 $pflip_0$ = ~0.00	$pflip_{-4}$ = 0.97 $pflip_{-2}$ = 0.01 $pflip_0$ = ~0.00	$pflip_{-4}$ = 0.99 $pflip_{-2}$ = 0.38 $pflip_0$ = ~0.00
Sigma = 0.5	$pflip_{-4}$ = 0.50 $pflip_{-2}$ = 0.02 $pflip_0$ = ~0.00	$pflip_{-4}$ = 0.99 $pflip_{-2}$ = 0.50 $pflip_0$ = ~0.00	$pflip_{-4}$ = 0.99 $pflip_{-2}$ = 0.98 $pflip_0$ = ~0.00
Sigma = 0.6	$pflip_{-4}$ = 0.92 $pflip_{-2}$ = 0.17 $pflip_0$ = ~0.00	$pflip_{-4}$ = 0.99 $pflip_{-2}$ = 0.92 $pflip_0$ = ~0.00	$pflip_{-4}$ = 0.99 $pflip_{-2}$ = 0.99 $pflip_0$ = ~0.00

In the above table, notice that for J = 3, sigma = 0.5, the values of $pflip_{-2}$ and $pflip_{-4}$ are very high while the values of $pflip_0$ is almost zero. This means that the pixel will very likely flip when surrounded by 3 or 4 opposing pixels, but very unlikely to flip when surrounded by 2 opposing pixels. Also, notice that (J = 2, sigma = 0.6) and (J = 3, sigma = 0.6) are also equally good values of parameters to use. But we used **(J = 3, sigma = 0.5).**

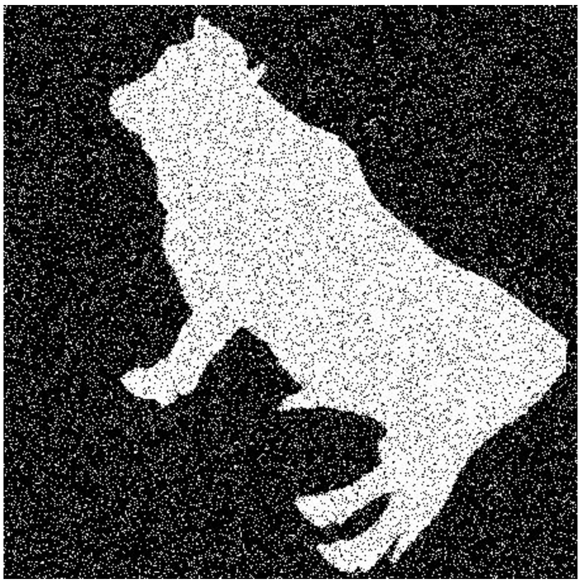
Similarly, we used **(W = 3, c = 0.5)** as the parameters for variational inference.

Using the parameters above, and the equations outlined the Assignment given, we ran gibbs sampling and variational inference on the noisy images. Examples of the results obtained can be found in the results folder along with this report, as well as in the section below.

Miscellaneous notes:

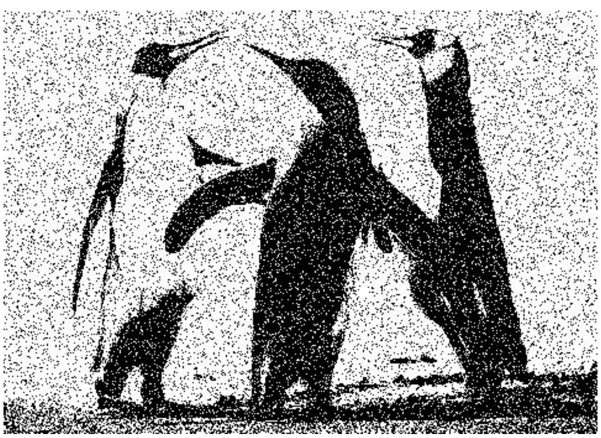
1) The results of Gibbs sampling and variational inference are almost identical, since the underlying equations follow the same Ising Model along with the same parameters for strength of local evidence and coupling. Also, the same number of iterations (5) were used in both cases
2) All the functions used in the implementation are self-written. Except '*norm.pdf*' which is taken from the '*scipy.stats*' python library.
3) Instructions to run the denoising script again can be found in the README.md file along in the same folder as this report.

Gibbs Sampling - Results:



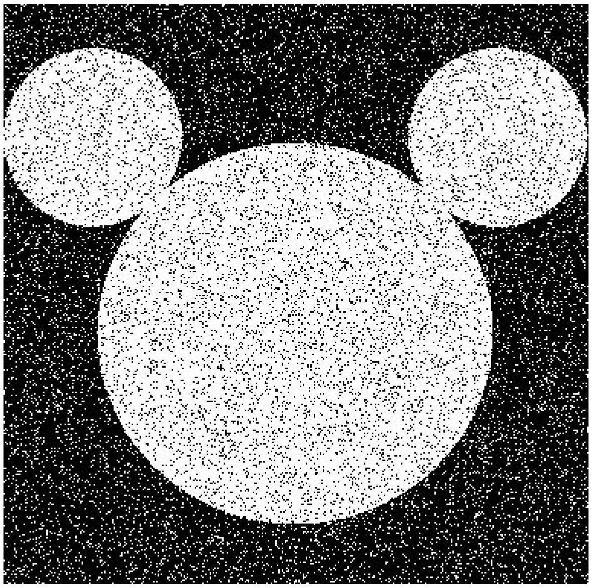
Original image

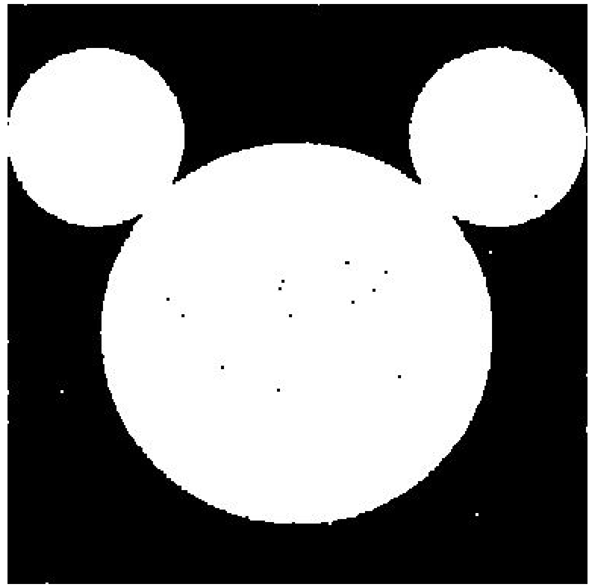
5 iterations of Gibbs Sampling



Original image

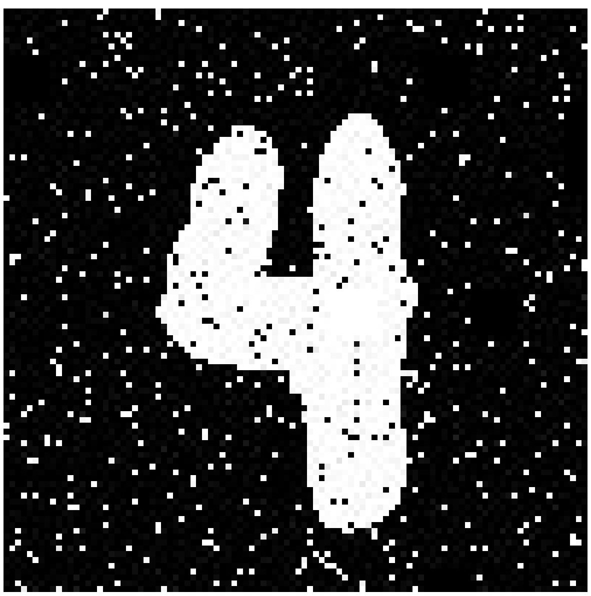
5 iterations of Gibbs Sampling

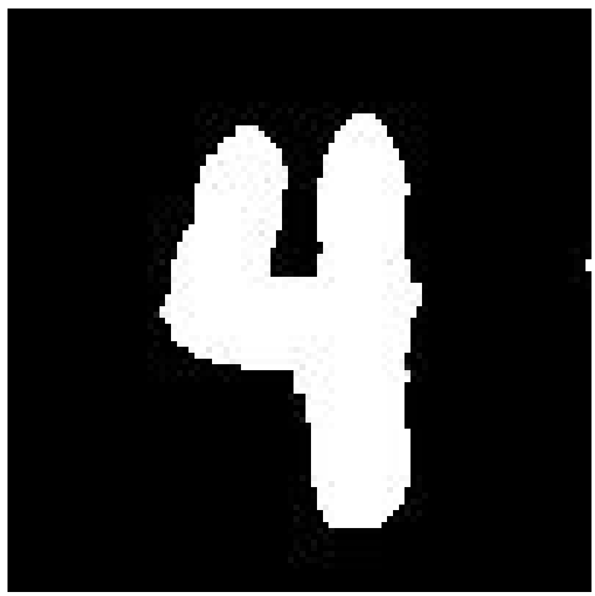




Original image

5 iterations of Gibbs Sampling

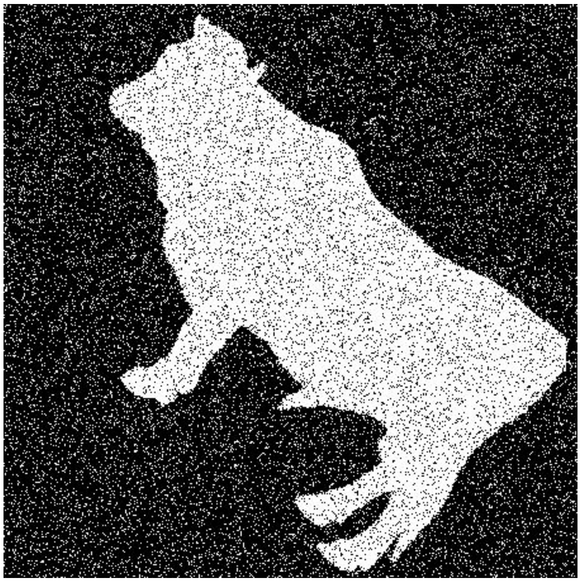




Original image

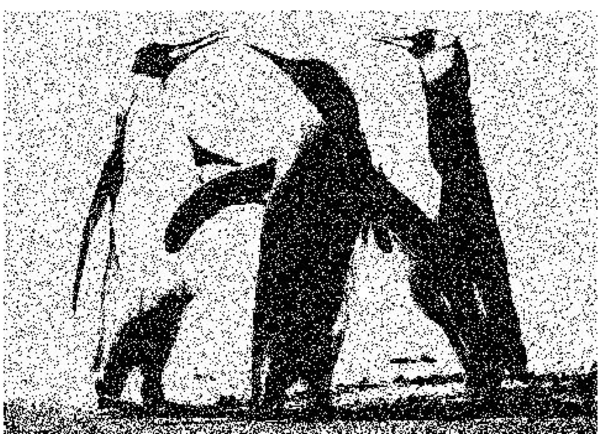
5 iterations of Gibbs Sampling

Variational Inference - Results:



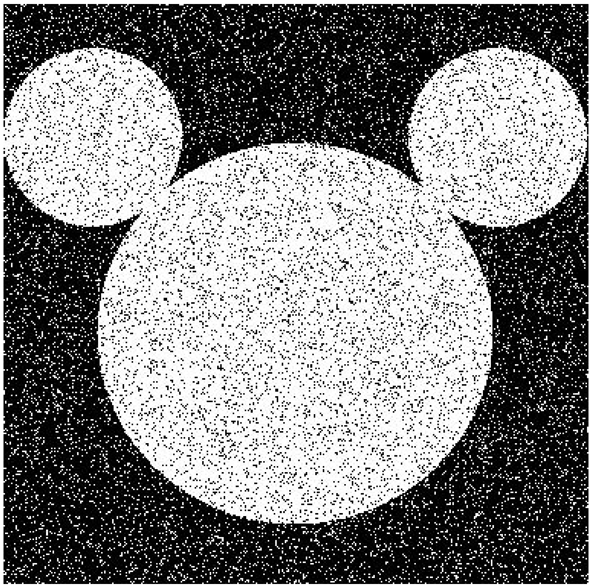
Original image

5 iterations of Variational Inference

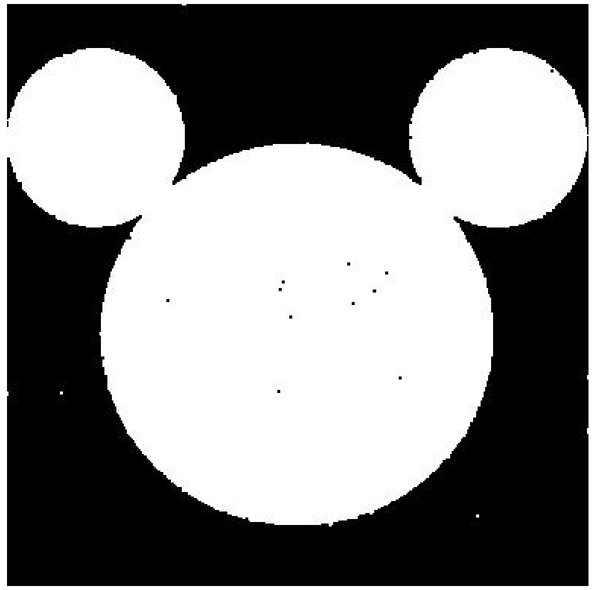


Original image

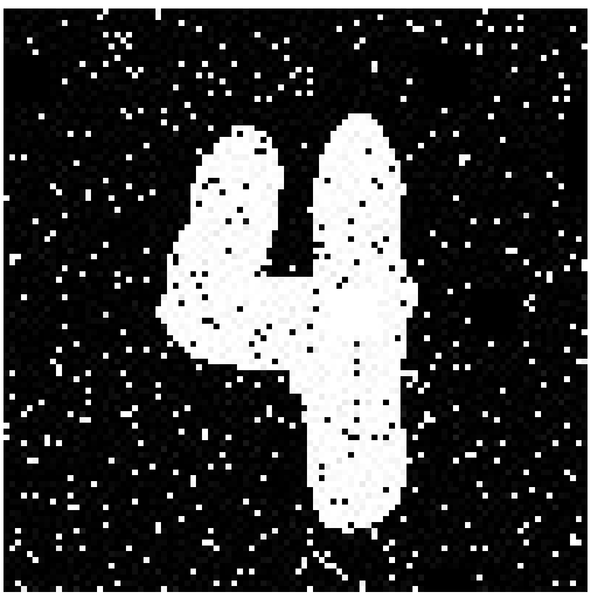
5 iterations of Variational Inference



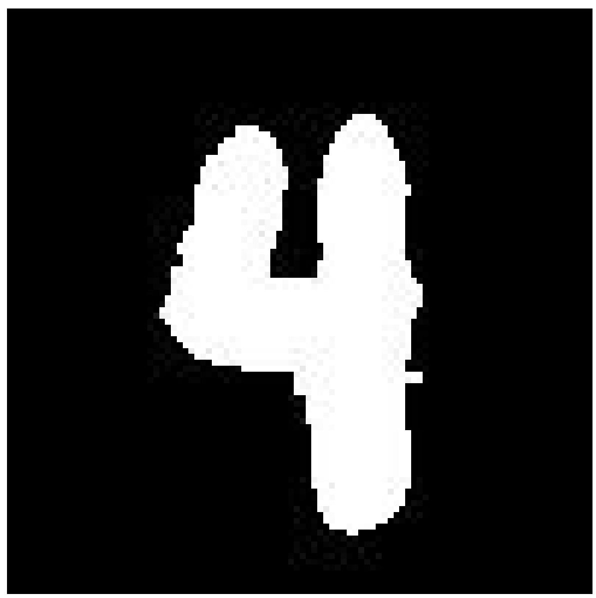
Original image



5 iterations of Variational Inference



Original image



5 iterations of Variational Inference

Image Segmentation using EM Algorithm

Given a set of images, our aim here is to decouple the foreground from the background. In order, to achieve this, we use the EM algorithm to obtain a Gaussian mixture of 2 components, where we alternatively estimate the log-likelihood using the parameters and then compute the value of parameters by maximising the log-likelihood. We assume that the reader is familiar with EM algorithm and Gaussian Mixture Models and we shall proceed to describe the salient features of our implementation only.

Feature-set

We used 5 features to define a 5-dimensional Gaussian. The features are (x, y, L, a, b). Where (x,y) are the location coordinates and (L, a, b) are the color coordinates in CIELAB space.

Initialisation of EM:

The choice of the mean of each gaussian distribution is crucial as the EM algorithm is highly sensitive to initialisation.

We noticed that in the data provided, the subjects seem to be fairly in the middle and hence, we opted to initialise the segments as two 'windows': the **subject** where ¼ < x < ¾ and ¼ < y < ¾, and the **background** where x < ¼, x > ¾ and y < ¼ and y > ¾ as shown below in Fig1. This is done by checking if the values for data[i,0] and data[i,1] fall below the half range. If yes, all these are added to a list and the ones that don't are added to a separate list. The formula used for the calculation of half range is as given below:

```
half_range = max(data[:,0])/2
```

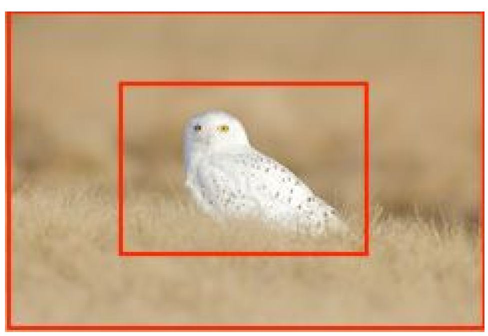


Fig1. Segmentation of image to centre and background.

For initialisation of the means of the 2 Gaussians, we took the means of the (x, y, L, a, b) values of the 2 windows.

Other initialisation methods that we did not use:

- Random initialisation of values results in convergence to local optima and thus, results in poor segmentation
- Usage of K-means (or K-means++) for initialisation slows down the computation unless an existing library such as the one provided by sklearn is used.

- Hence, we opted for the ‘windowed’ approach to initialisation which gave a good convergence.

Pre-processing

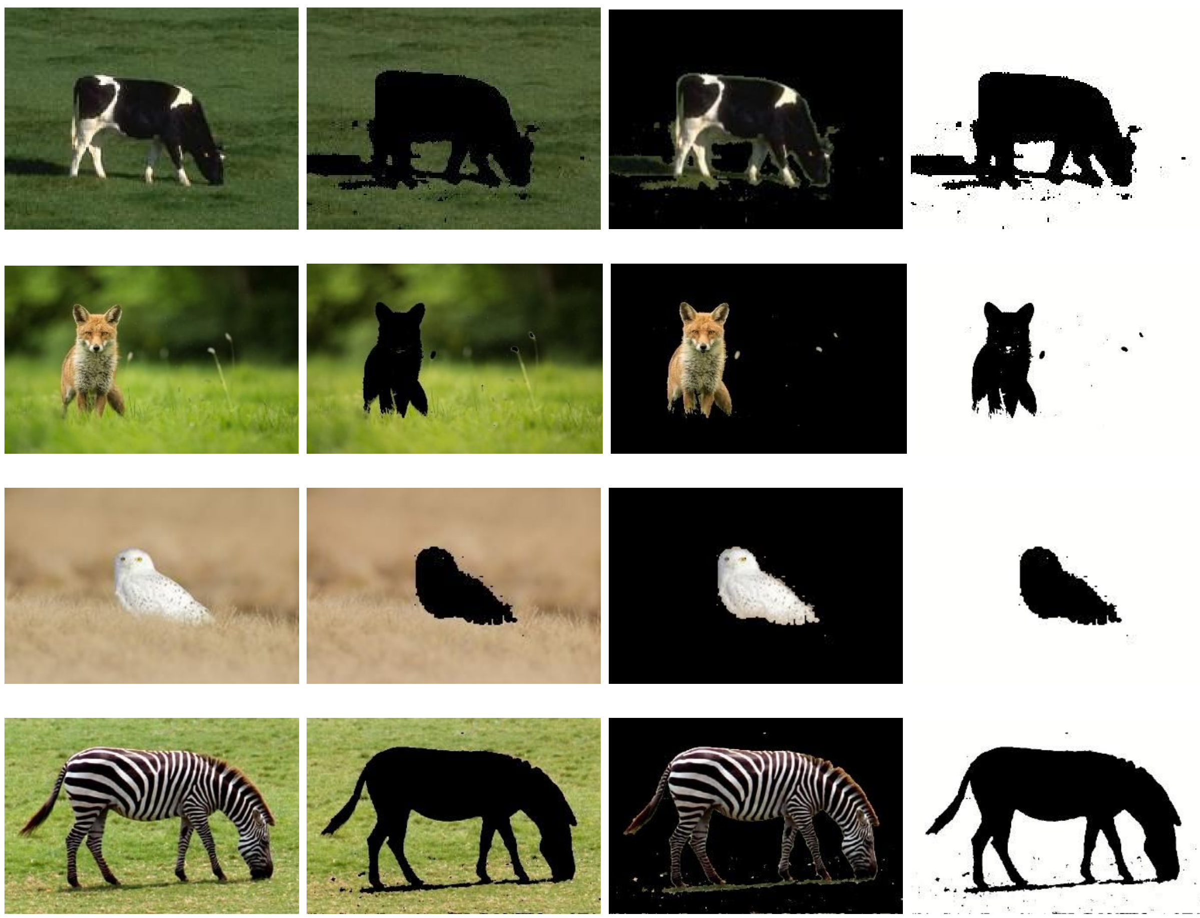
The feature-set is first standardised using the following formula:

```
(data[:,i] - np.mean(data[:,i]))/np.std(data[:,i])
```

Ie.- we subtract the means and divide by standard deviation. This is important to ensure that the data lies on the same scale. Convergence is faster when the data is standardised.

Results

The outputs obtained from our implementation are shown below and can be found in the results folder.



Miscellaneous notes:

1) EM algorithm is highly sensitive to initialisation, and is not used in state-of-the-art segmentation task, unless very special structural details about the images are known.
2) All the functions used in the implementation are self-written. Except ‘*multivariate_normal.pdf*’ which is taken from the ‘*scipy.stats*’ python library.