

Assignment 3 Report

CS-726: Advanced Machine Learning

Deeptanshu Malu Deevyanshu Malu Neel Rambhia

1 Task 0

The output provided on running the program is:

Using device: cuda

--- Model Architecture ---

```
EnergyRegressor(  
  (net): Sequential(  
    (0): Linear(in_features=784, out_features=4096, bias=True)  
    (1): ReLU(inplace=True)  
    (2): Linear(in_features=4096, out_features=2048, bias=True)  
    (3): ReLU(inplace=True)  
    (4): Linear(in_features=2048, out_features=1024, bias=True)  
    (5): ReLU(inplace=True)  
    (6): Linear(in_features=1024, out_features=512, bias=True)  
    (7): ReLU(inplace=True)  
    (8): Linear(in_features=512, out_features=256, bias=True)  
    (9): ReLU(inplace=True)  
    (10): Linear(in_features=256, out_features=128, bias=True)  
    (11): ReLU(inplace=True)  
    (12): Linear(in_features=128, out_features=64, bias=True)  
    (13): ReLU(inplace=True)  
    (14): Linear(in_features=64, out_features=32, bias=True)  
    (15): ReLU(inplace=True)  
    (16): Linear(in_features=32, out_features=16, bias=True)  
    (17): ReLU(inplace=True)  
    (18): Linear(in_features=16, out_features=8, bias=True)  
    (19): ReLU(inplace=True)  
    (20): Linear(in_features=8, out_features=4, bias=True)  
    (21): ReLU(inplace=True)  
    (22): Linear(in_features=4, out_features=2, bias=True)  
    (23): ReLU(inplace=True)  
    (24): Linear(in_features=2, out_features=1, bias=True)  
  )  
)
```

Loading dataset from ../A4_test_data.pt...

Dataset loaded in 0.17s. Shape: `x=torch.Size([100000, 784])`, `energy=torch.Size([100000, 1])`

--- Test Results ---

Loss: 288.1554

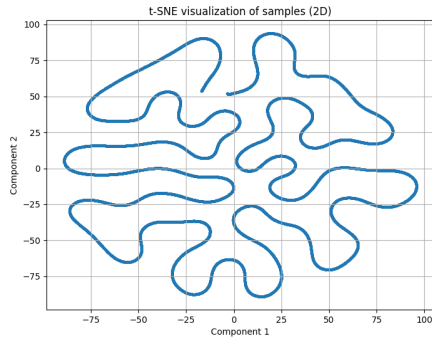
--- Script Finished ---

2 Task 1

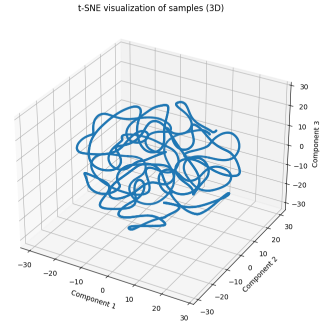
10000 samples were taken in total using each algorithm. 2000 samples were used for burn-in and 8000 samples were used for sampling.

Sampling Algorithm	Burn-in time (s)	Sampling time (s)	Total time (s)
Algorithm 1	19.37	71.23	90.60
Algorithm 2	8.19	32.53	40.72

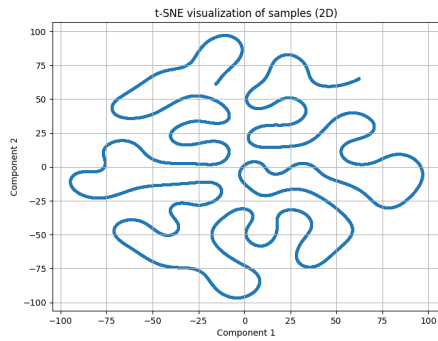
Table 1: Performance comparison of sampling algorithms for 10,000 samples.



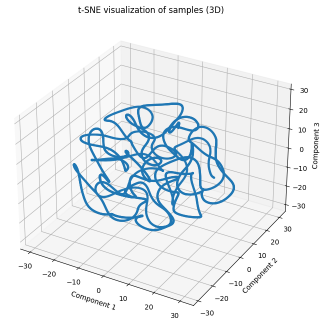
Algorithm 1 t-SNE 2d



Algorithm 1 t-SNE 3d



Algorithm 2 t-SNE 2d



Algorithm 2 t-SNE 3d

Figure 1: t-SNE plots for the two sampling algorithms.

3 Task 2

n is the number of samples taken from the Branin-Hoo function.

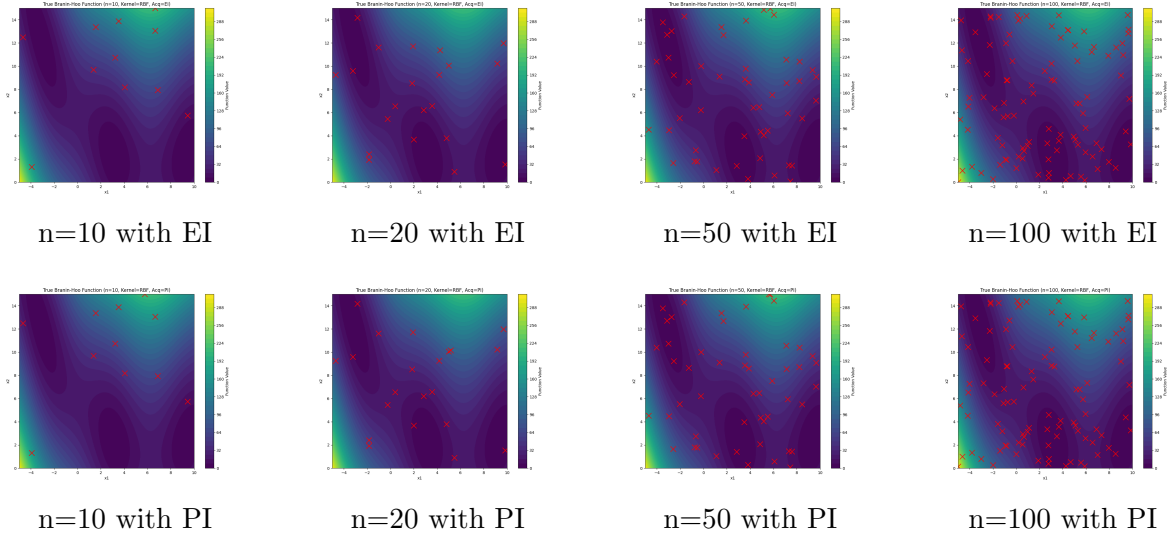


Figure 2: RBF kernel true function with different acquisition functions.

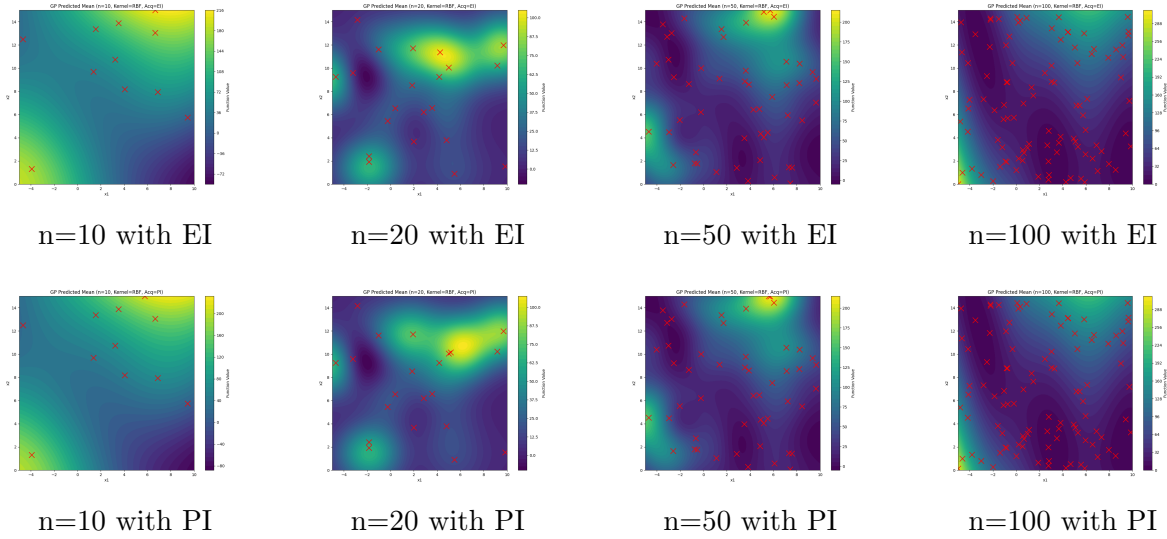


Figure 3: RBF kernel GP mean with different acquisition functions.

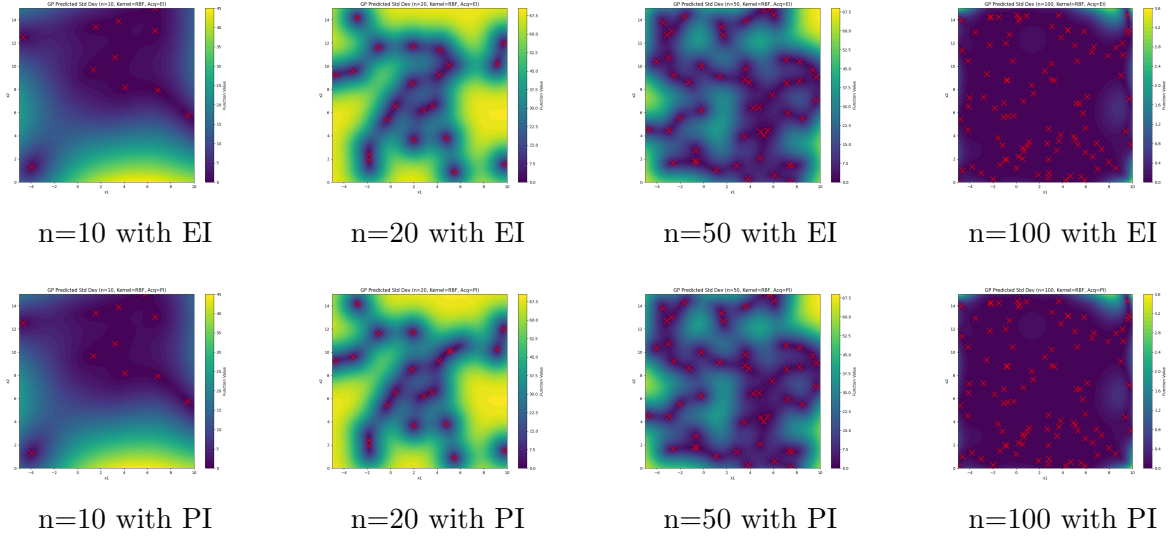


Figure 4: RBF kernel GP standard deviation with different acquisition functions.

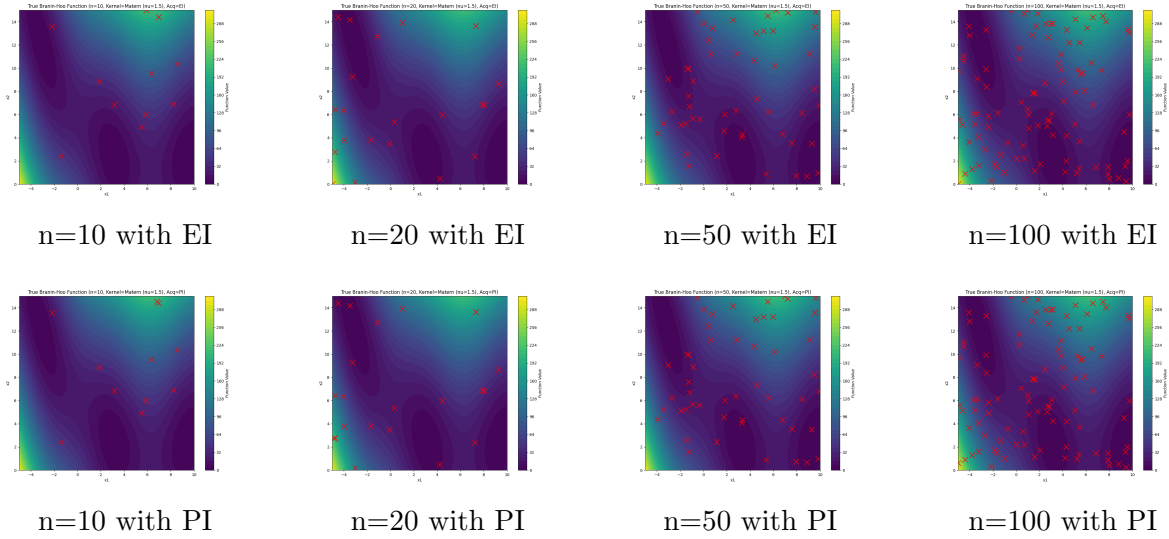


Figure 5: Matern kernel true function with different acquisition functions.

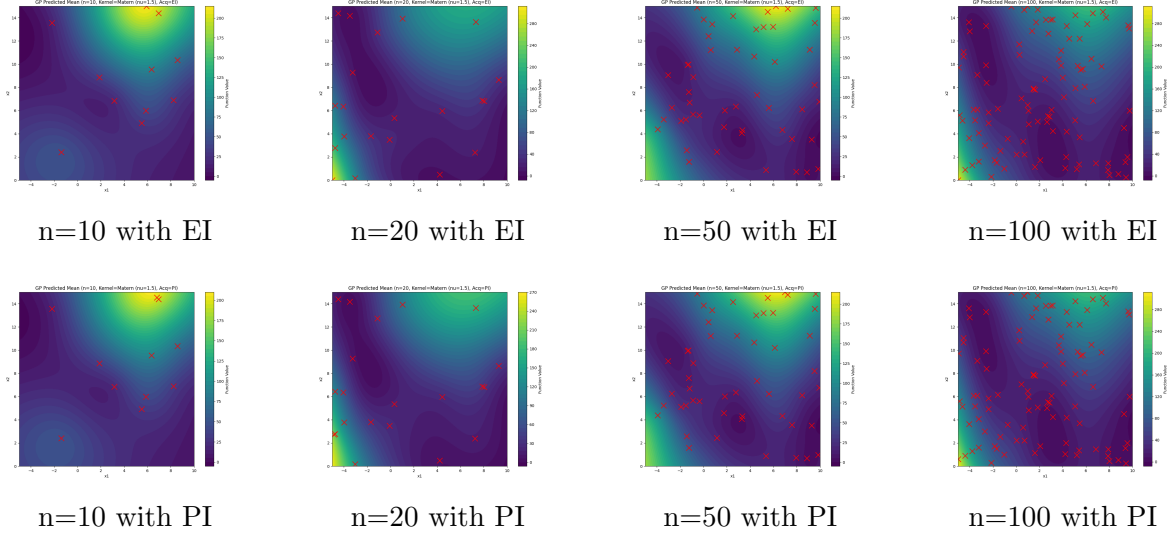


Figure 6: Matern kernel GP mean with different acquisition functions.

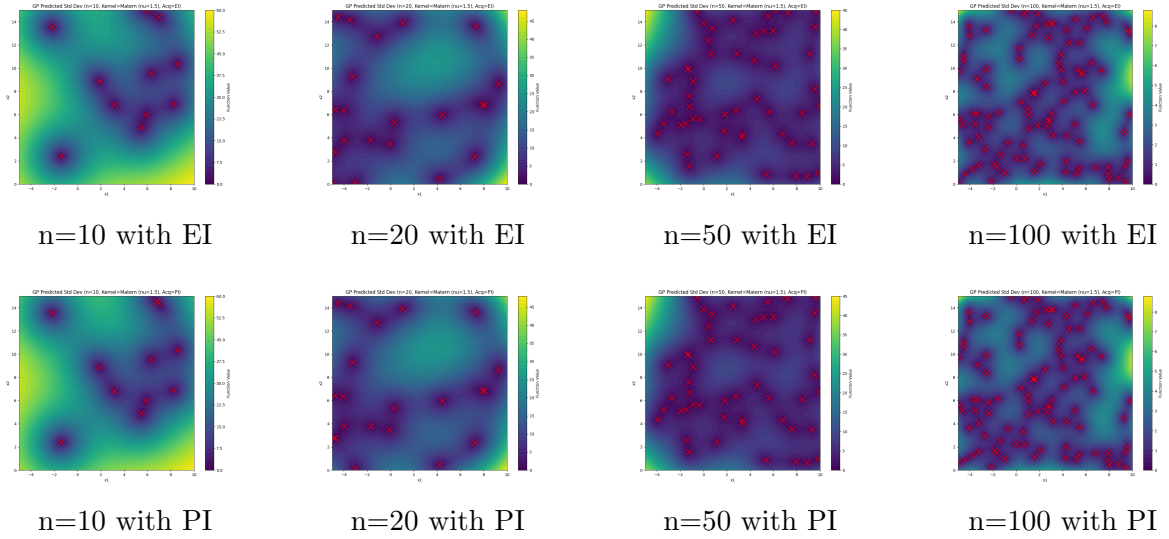


Figure 7: Matern kernel GP standard deviation with different acquisition functions.

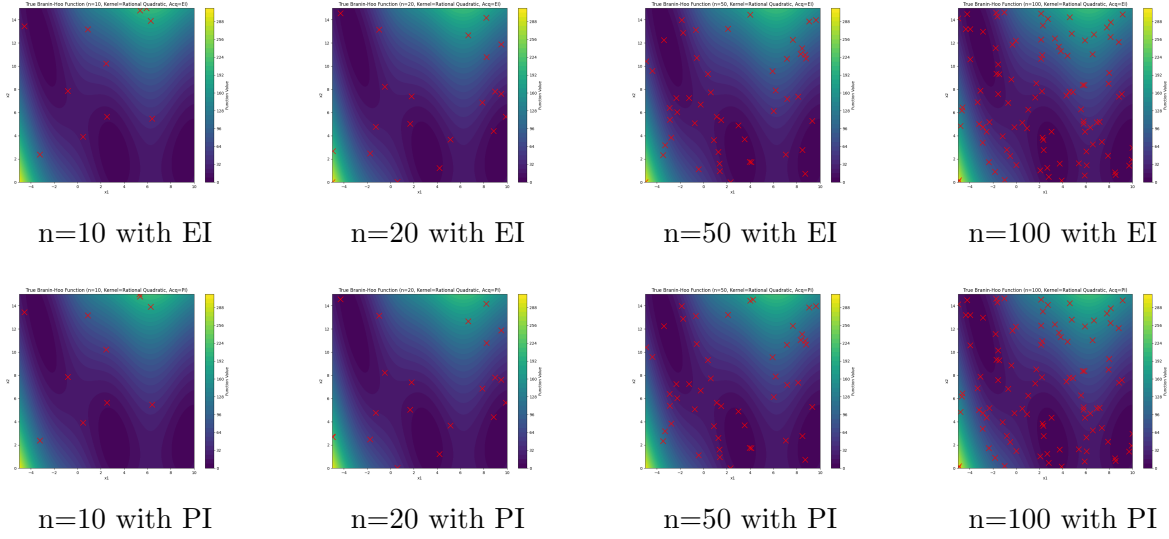


Figure 8: Rational Quadratic kernel true function with different acquisition functions.

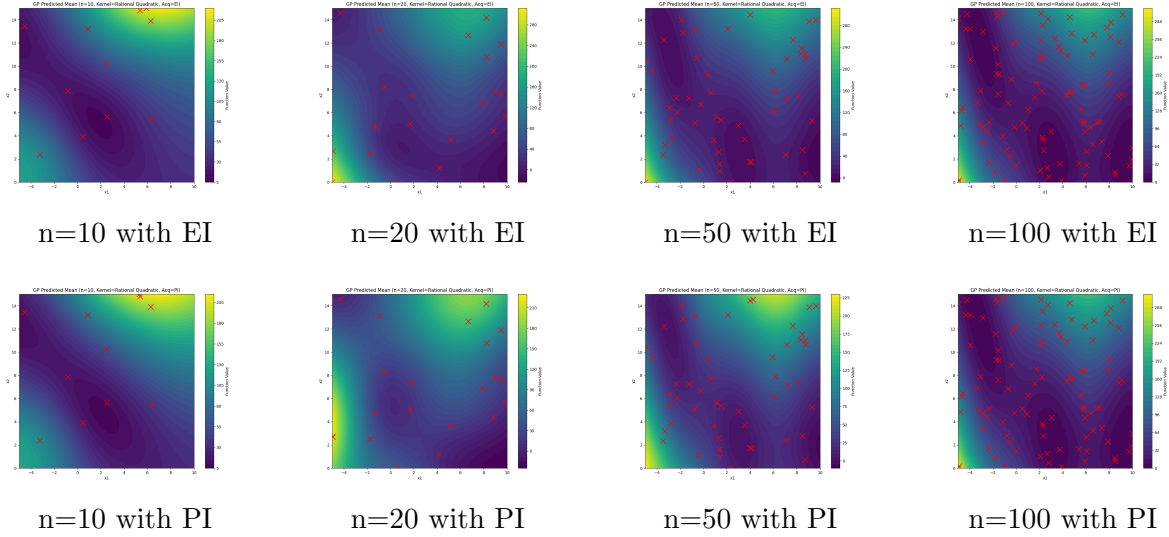


Figure 9: Rational Quadratic kernel GP mean with different acquisition functions.

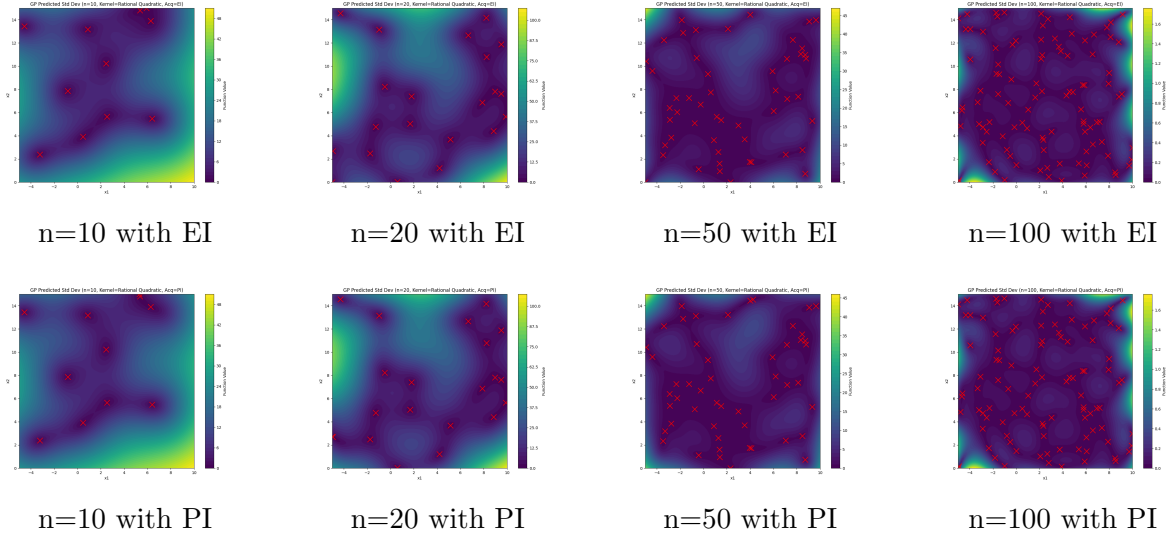


Figure 10: Rational Quadratic kernel GP standard deviation with different acquisition functions.

Observations

1. Effect of Sample Size (n):

- As n increases, the GP mean becomes more accurate and closely resembles the true function.
- The standard deviation decreases, indicating improved confidence in predictions.

2. EI vs. PI:

- **EI** promotes better exploration, especially at smaller n , by sampling more diverse regions.
- **PI** is more exploitative and often clusters samples, potentially leading to suboptimal local solutions.

3. Kernel Behavior:

- **RBF Kernel:** Produces smooth, globally consistent approximations. Performs well across all n .
- **Matern Kernel:** Shows more localized variations; slower to converge but captures fine-grained features.
- **Rational Quadratic Kernel:** Balances smoothness and local adaptability. Strong performance even at low n .

4. Uncertainty Patterns:

- Higher uncertainty (GP std dev) is seen at the edges or unsampled regions.
- These areas shrink as n increases, particularly with EI-based acquisition.

Contributions

- **Deeptanshu Malu:**

1. Formulated the trie data structure design for Task 1.
2. Coded the multi head decoding algorithm for Task 2.

- **Deevyanshu Malu:**

1. Coded the trie data structure and the constrained decoding algorithm for Task 1.
2. Coded the multi head decoding algorithm for Task 2.

- **Neel Rambhia:**

1. Completed all decoding algorithms in Task 0.
2. Coded the single head decoding algorithm for Task 2.

Acknowledgements

- We have also used Copilot for faster coding and not for direct logic.
- Used ChatGPT to generate the trie data structure but filled in the logic ourselves.