CSE849 Project 4

Javen W. Zamojcin
zamojci1@msu.edu

2025, April 30

1 Task 1: Unconditional Generation

The Unconditional Generation task of this project comprised of implementing and training a denoiser MLP model, and a method to unconditionally sample using this trained denoiser.

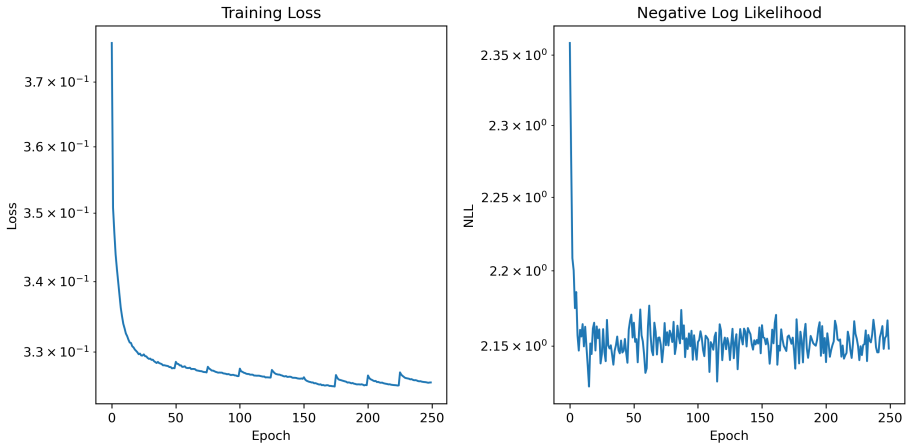


Figure 1: Denoiser Model Training History

In Figure 1, we observe both the training loss (MSE) and Negative Log Likelihood (NLL) metric history over the 250 epoch course of training and sampling the denoiser model. In the training loss figure, we observe local spikes which are reflective of the 25 epoch interval data mixing, where the training dataset samples (2,500,000 total) are precomputed with fresh noise. After each training epoch, we unconditionally sampled (2000 samples over 500 steps) using the denoiser model, and the NLL is calculated as a performance metric for the sampling.

Figure 2 demonstrates the final unconditional sampling (5000 samples over 500 steps) after 250 training epochs. The NLL for this final sampling was around 2.15, and shows a strong fidelity to the original image of these five states.

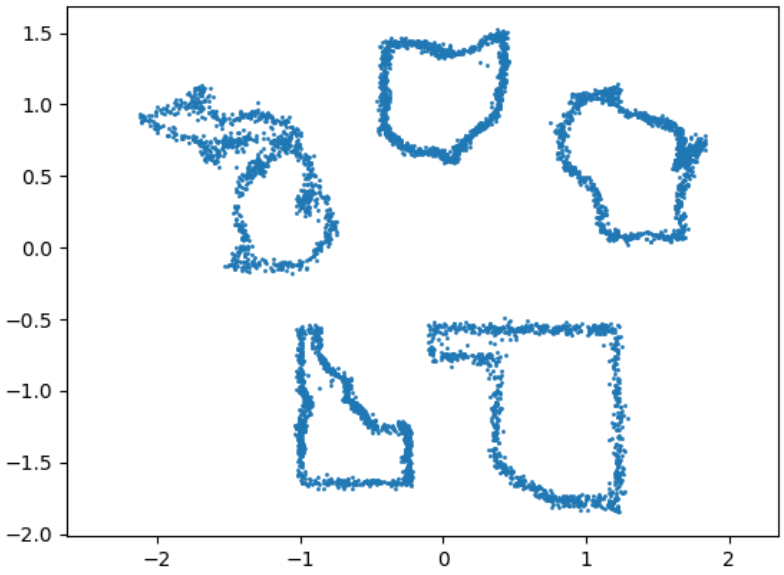


Figure 2: Unconditional Final Generation

The following are the hyperparameters used for the Unconditional Generation denoiser training process:

```
batch_size = 10_000
n_epochs = 300
lr = 1e-3
weight_decay = 1e-7
n_steps = 500
refresh_interval = 25
save_interval = 10

denoiser = MLP(input_dim=3, output_dim=2, hidden_layers=[256, 256, 256, 256])
mse_loss = nn.MSELoss()
optimizer = torch.optim.AdamW(denoiser.parameters(), lr=lr, weight_decay=weight_decay)
scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=2, gamma=0.99)
```

2 Task 2: Conditional Generation

2.1 Classifier Training

For the task of Conditional Generation, the first sub-task was to implement and train a classifier MLP model to accurately predict the state label given the set of noisy points at any given timestep.

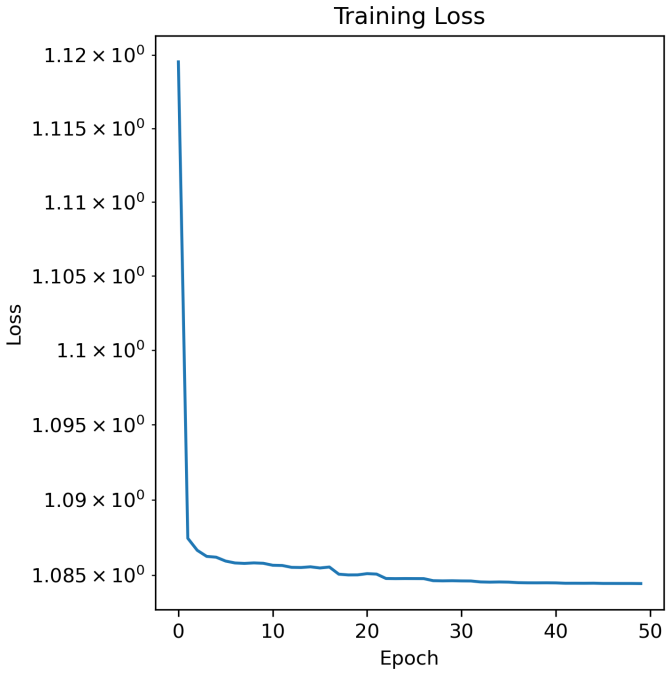


Figure 3: Classifier Model Training History

In Figure 3, we observe the training loss (Cross-Entropy) metric history over the 50 epoch course of training the classifier, which aims to correctly classify each of the five states given an instance of their noisy data. Figure 4 demonstrates a qualitative evaluation of the trained classifier models predictions. We observe that the classifier model circumscribes the various states fairly well, with the notable exception being the Oklahoma/Idaho border marginally misclassifying part of Oklahoma. This likely is due to these two states being located so close together, and would require further hyperparameter experimentation to fix.

The following are the hyperparameters used for the Conditional Generation classifier training process:

```
n_steps = 500
batch_size = 10_000
n_epochs = 50
```

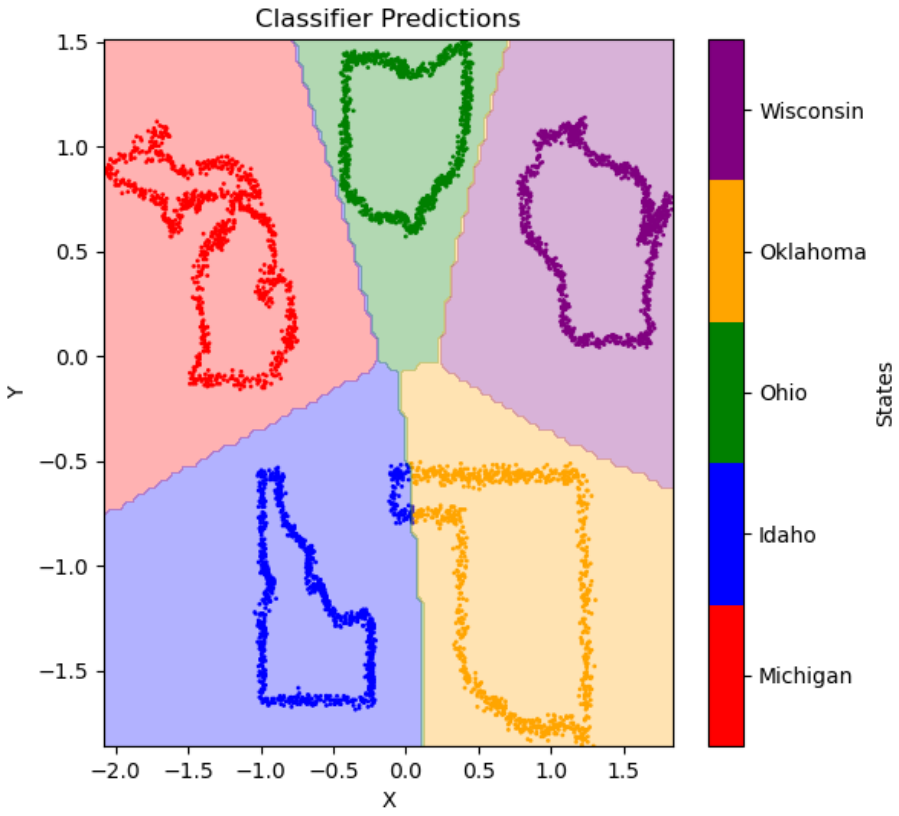
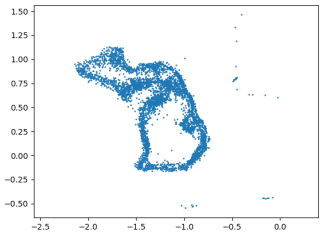


Figure 4: Classifier Model Predictions

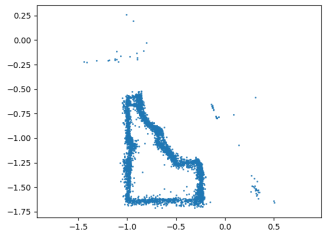
```
lr = 0.001
weight_decay = 1e-4
refresh_interval = 100

classifier = MLP(input_dim=3, output_dim=5, hidden_layers=[100, 200, 500])
ce_loss = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(classifier.parameters(), lr=lr, weight_decay=weight_decay)
scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(
    optimizer,
    mode='min',
    factor=0.5,
    patience=3,
)
```

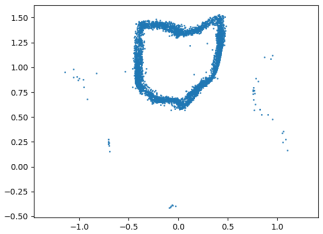
2.2 Conditional Generation



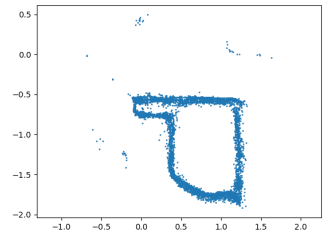
(a) label-0



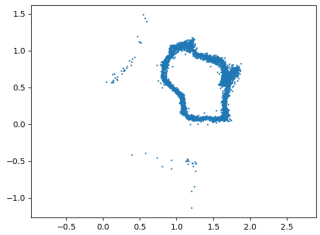
(b) label-1



(c) label-2



(d) label-3



(e) label-4

Figure 5: Conditionally Generated Plots

Once the denoiser and classifier models were successfully trained, the Conditional Generation task could be completed. This involved restoring/loading the two pretrained models, and modifying the sampling method for conditional generation. For each of the five state labels, the sampling involved 5000 noisy samples across 500 steps. For each step, the noise is predicted from the previous step's noisy points using the denoiser like before, but the key to conditional sampling is calculating the gradients of the log-softmax classifier predictions w.r.t. the predicted noisy input data for the target label, and using these gradients to shape the generation toward the target label. No training is involved in this

step of the process.

In Figure 5, we observe the final conditionally generated samples (5000 samples over 500 steps) for each of the respective five state labels (Michigan, Idaho, Ohio, Oklahoma, Wisconsin). The fidelity is fairly strong for each of the labels, but each also contains noisy outlier samples. I suspect this is likely due to the imperfect classification model. The following is the list of final NLLs for each of the five conditionally generated plots:

```
Label 0, NLL: 2.1386
Label 1, NLL: 1.9886
Label 2, NLL: 2.0409
Label 3, NLL: 2.4367
Label 4, NLL: 2.131
```