Technical Report: MNIST Diffusion Experiments on Purdue Gautschi

PurdueHPC_Codex Workflow

February 24, 2026

1 Scope and Environment

This report summarizes end-to-end execution of diffusion model experiments on Purdue RCAC Gautschi, including environment setup, Slurm execution, model details, and resulting artifacts. All commands and scripts were executed under:

- Scratch workspace: `/scratch/gautschi/rmaulik/codex_test`
- Source repository: `PurdueHPC_Codex`
- Account: `rmaulik`
- Partition: `ai`

No credentials or authentication secrets are stored in this report.

2 Allocation and Job Accounting

Current allocation snapshot from `slist` on Gautschi:

- AI partition GPU-hour balance: **43,798.5**
- CPU partition balance: **0**

Completed experiment jobs:

Job ID	Name	State	Elapsed (s)	GPUs
8109439	mnist_ddpm	COMPLETED	41	1
8109477	mnist_ddpm_long	COMPLETED	99	1
8109577	mnist_ddpm_long	COMPLETED	411	1

Estimated direct GPU-hours from these three runs:

$$\text{GPU-hours} = \frac{41 + 99 + 411}{3600} \times 1 = 0.153 \text{ GPU-hours (approx.)}$$

3 Diffusion Model Formulation

The training script implements a DDPM-style denoising objective over MNIST ($x_0 \in [-1, 1]^{1\times28\times28}$).

3.1 Forward Diffusion

Define variance schedule $\{\beta_t\}_{t=1}^{T}$ with

$$\beta_t \in [10^{-4}, 2 \times 10^{-2}], \quad \alpha_t = 1 - \beta_t, \quad \bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s.$$

The forward Markov transition is

$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t} x_{t-1}, \beta_t I),$$

with closed form sample

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon, \quad \epsilon \sim \mathcal{N}(0, I).$$

3.2 Reverse Model and Objective

The denoiser network $\epsilon_\theta(x_t, t)$ predicts injected noise. Reverse mean step used in sampling:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right).$$

At training time, the objective is standard noise-prediction MSE:

$$\mathcal{L}(\theta) = \mathbb{E}_{x_0, \epsilon, t} \left[\left\| \epsilon - \epsilon_\theta\big(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, t \big) \right\|_2^2 \right].$$

4 Network Architecture

The current model is a residual U-Net style denoiser with sinusoidal time embeddings:

- Stem: Conv $1 \to 64$
- Down path: residual blocks at 64 and 128 channels with strided downsampling $(28 \to 14 \to 7)$
- Bottleneck: two residual blocks at 256 channels
- Up path: transposed convolutions + skip concatenation + residual blocks $(7 \to 14 \to 28)$
- Output: GroupNorm + Conv $64 \to 1$
- Optimizer: AdamW, gradient clipping (max norm 1.0)

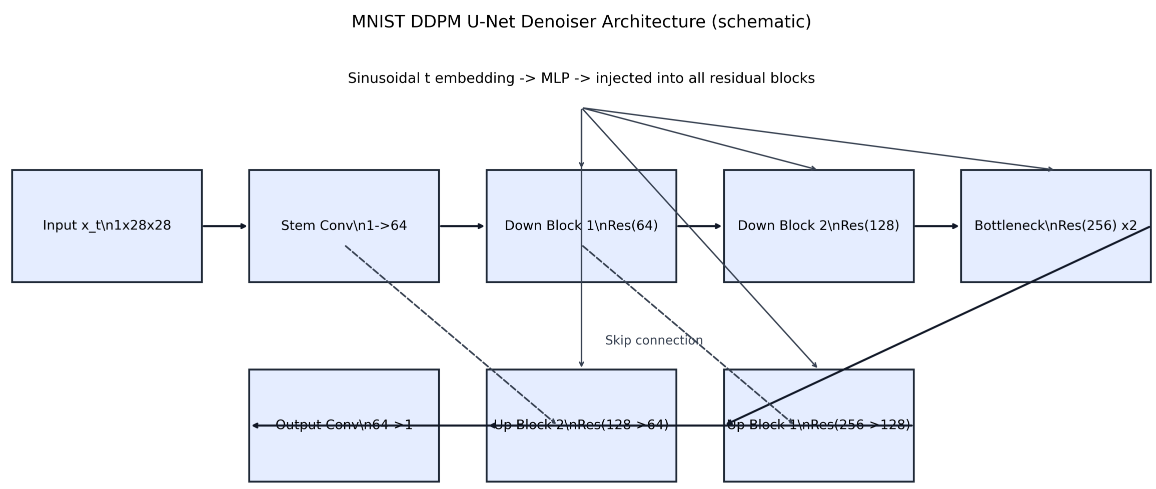

Figure 1: U-Net denoiser schematic generated by the training pipeline.

5 Latest Long Run Configuration

The latest long run (job `8109577`) used:

- Epochs: 80
- Diffusion steps (T): 300
- Batch size: 128
- Slurm resources: 1 H100 GPU, 14 CPUs, partition `ai`, walltime 4h

From `metrics.json` (run `long_8109577`):

- Total steps: 37,520
- Final epoch mean loss: 0.0362516
- Best epoch mean loss: 0.0360608
- Device: CUDA

6 Results and Artifacts

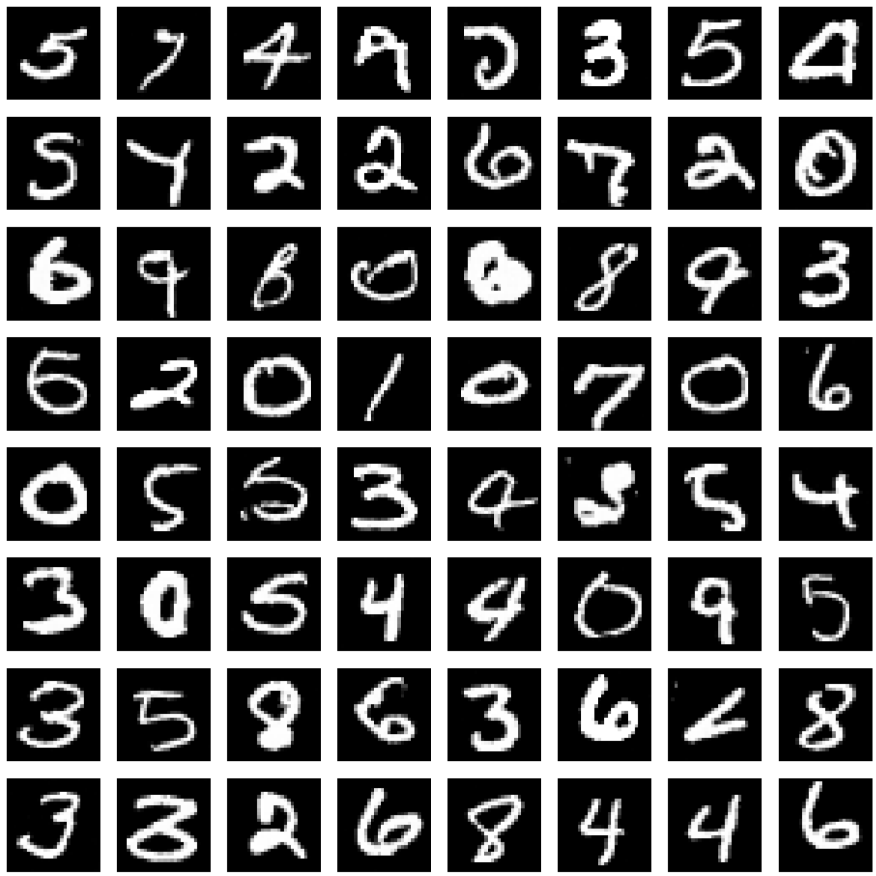

Figure 2: Generated MNIST samples from the latest completed run.

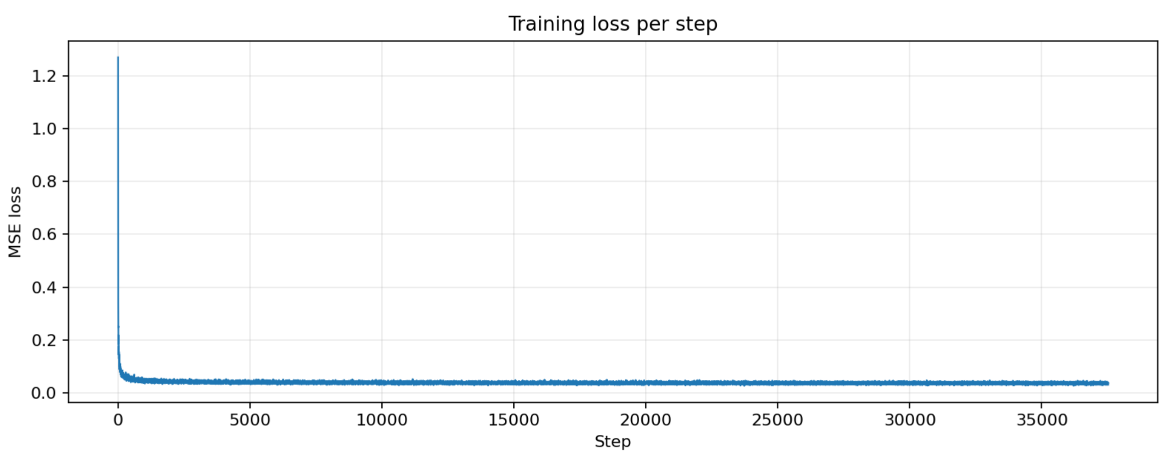

Figure 3: Per-step training loss trajectory.

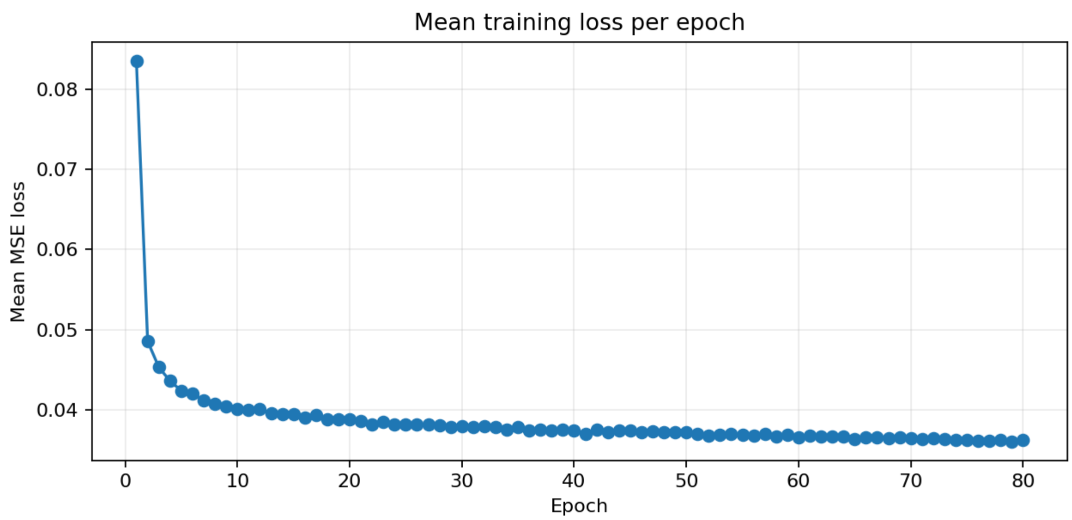

Figure 4: Per-epoch mean loss trajectory.

7 Posterior Sampling with Partial Observations

We added a post-training posterior sampling routine to `mnist_diffusion.py` that performs likelihood-guided reverse diffusion under partial pixel observations. The latest implementation includes:

- Noise-aware likelihood variance: $\sigma_t^2 = \sigma_y^2 + c(1 - \bar{\alpha}_t)$
- Annealed guidance schedule: weak guidance at high noise, stronger guidance near final denoise steps
- Data-consistency projection on observed pixels (in both $\hat{x}_0$ and sampled states)

For this experiment:

- Observed fraction: 70% of pixels (30% occluded, random mask)
- Target digit class: 7
- Guidance scale: 1.5
- Guidance min fraction / power: 0.25 / 1.5
- Likelihood noise scale: 0.1
- Noise-aware coefficient: 0.05

Validated Gautschi posterior job:

- Job ID: `8112324`
- State: `COMPLETED`, exit code `0:0`
- Elapsed: `00:00:27`
- Run tag: `posterior_70pct_dpsplus_8112324`

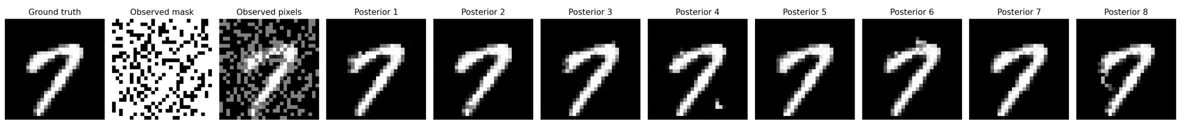

Figure 5: Improved DPS conditioning setup: ground truth, observed mask (70% observed), observed pixels, and posterior draws.

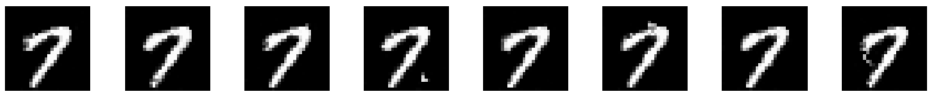

Figure 6: Posterior samples with noise-aware likelihood, guidance annealing, and projection-based data consistency.

8 Live Dashboard Operation

The workflow now maintains a root live dashboard at:

- `outputs/dashboard.html`

This dashboard follows the latest run via:

- `outputs/LATEST_RUN.txt`
- `outputs/current` symlink

and supports:

- periodic refresh during active runs,
- automatic stop of refresh when status reaches `completed`,
- image zoom controls for plots/schematics.

9 Reproducibility Notes

Primary scripts tracked in Git:

- `mnist_diffusion.py`
- `submit_mnist_diffusion_long.slurm`
- `serve_dashboard.sh`

To regenerate this report PDF locally:

```
cd PurdueHPC_Codex/report
latexmk -pdf -interaction=nonstopmode -halt-on-error \
  -output-directory=../output/pdf gautschi_diffusion_report.tex
```