Advanced Machine Learning (CS60073)

Assignment 3: Generative Modeling

Total Marks: 150

Submission Instructions

- Submit a **single Jupyter notebook**: per part.
- Add a brief **Markdown explanation before every code cell** describing what the cell does.
- Use **CPU only**. Set global random seed `seed = 1234` for NumPy and PyTorch.
- For Part B use **MNIST** from `torchvision`. For Part C use **Tiny Shakespeare** (character-level).
- Use the exact model sizes, hyperparameters, and evaluation settings specified below.
- Implementations must match the specified sizes and hyperparameters unless explicitly stated otherwise.
- For MNIST + GAN: map images from $[0, 1]$ to $[-1, 1]$ before feeding to D; for VAE keep in $[0, 1]$ with Sigmoid output.
- For Part A, numerical stability tricks (log-sum-exp, Cholesky) are allowed; do not change target/proposal definitions.

Part A – Sampling Methods (Total: 50 marks)

Goal: Understand and compare classical sampling methods for approximating complex probability distributions.

Settings (Part A)

- Use double precision (`float64`) for all probability computations.
- Random seed: **1234**.
- All contour plots use a mesh over $x, y \in [-6, 6]$ with resolution 200×200.

1) Define the target distribution (10 marks)

Target: 2D Gaussian Mixture with 3 components

$$p(x) = \sum_{k=1}^{3} \pi_k \mathcal{N}(x \mid \mu_k, \Sigma_k).$$

Weights: $\pi = [0.4, 0.35, 0.25]$.
Means: $\mu_1 = (0, 0)$, $\mu_2 = (3, 3)$, $\mu_3 = (-3, 3)$.
Covariances: $\Sigma_1 = \begin{bmatrix} 1.0 & 0.8 \\ 0.8 & 1.5 \end{bmatrix}$, $\Sigma_2 = \begin{bmatrix} 0.6 & 0 \\ 0 & 0.6 \end{bmatrix}$, $\Sigma_3 = \begin{bmatrix} 1.2 & -0.6 \\ -0.6 & 1.2 \end{bmatrix}$.

Output: Plot the true density contour over the specified mesh.

2) Rejection Sampling (10 marks)

Proposal: $q(x) = \mathcal{N}(x \mid 0,\, \sigma^2 I)$ with $\sigma = 2.5$.
Envelope constant: fixed $c = 12$.
Sampling until $N = 10{,}000$ accepted samples.
Outputs: Acceptance rate; scatter of accepted samples overlaid on the true contour.

3) Importance Sampling (10 marks)

Proposal: same $q(x)$ as above. Draw $N = 50{,}000$ i.i.d. samples.
Weights: $w_i = \frac{p(x_i)}{q(x_i)}$.
Estimator for $f(x) = x_1^2 + \sin(x_2)$: $\hat{\mathbb{E}}[f] = \frac{\sum_i w_i f(x_i)}{\sum_i w_i}$. Effective sample size: $\text{ESS} = \frac{(\sum_i w_i)^2}{\sum_i w_i^2}$.
Outputs: Estimated mean, weighted sample variance, and ESS.

4) Metropolis–Hastings (10 marks)

Target: same mixture $p(x)$.
Random-walk proposal: $x' = x + \epsilon,\ \epsilon \sim \mathcal{N}(0, s^2 I)$.
Test step sizes: $s \in \{0.1,\, 0.5,\, 1.5\}$. For each s, run 50,000 iterations from $x_0 = (0, 0)$, burn-in 5,000, thin by 5.
Outputs: For $s = 0.5$, trace plot of x_1 and autocorrelation up to lag 50; acceptance rate for each s; brief comment on step size vs. mixing.

5) Gibbs Sampling (10 marks)

Use a correlated Gaussian target: $x \sim \mathcal{N}\left(\mu = \begin{bmatrix}0\\0\end{bmatrix},\ \Sigma = \begin{bmatrix}1 & 0.9\\0.9 & 1\end{bmatrix}\right)$. Conditionals: $x_1 \,|\, x_2 \sim \mathcal{N}(0.9\, x_2,\ 1 - 0.9^2)$, $\quad x_2 \,|\, x_1 \sim \mathcal{N}(0.9\, x_1,\ 1 - 0.9^2)$. Run 50,000 iterations from $(0, 0)$, burn-in 5,000, thin by 5.

Outputs: Compute ACF for x_1 up to lag 50, scatter of samples, and comment on mixing The **autocorrelation function (ACF)** of a sequence $\{x_t\}$ measures the correlation between samples separated by a lag k, defined as

$$\rho(k) = \frac{\text{Cov}(x_t,\, x_{t+k})}{\text{Var}(x_t)}.$$

A high $\rho(k)$ indicates that the samples are strongly correlated even k steps apart, whereas a low $\rho(k)$ implies weak dependence (better mixing).

Part B – VAE & GAN (Total: 60 marks)

Goal: Compare explicit-likelihood (VAE) and adversarial (GAN) generative modeling on MNIST (28×28 grayscale).

Settings (Part B)

- Splits: Train 50,000 / Val 10,000 / Test 10,000 (from torchvision MNIST: split the 60k train into 50k/10k).
- Inputs normalized to $[0, 1]$; for MLPs, flatten to 784.
- Optimizer: Adam, weight decay 0, batch size 128, epochs 10.

- Report losses per epoch and save fixed 10×10 sample grids each epoch.

1) Variational Autoencoder (10 marks)

Architecture (MLP)
Encoder (input 784): Linear(784→512) + ReLU; Linear(512→256) + ReLU; heads μ: Linear(256→16), $\log \sigma^2$: Linear(256→16).
Latent: $z = \mu + \sigma \odot \epsilon,\ \epsilon \sim \mathcal{N}(0, I_{16})$.
Decoder (input 16): Linear(16→256) + ReLU; Linear(256→512) + ReLU; Linear(512→784) + Sigmoid.
Loss: Reconstruction (binary cross-entropy, summed over pixels, averaged over batch) + KL to $\mathcal{N}(0, I)$.
Training: Adam(lr=1e-3, betas=(0.9,0.999)), epochs=10, batch=128.
Output: Plot reconstruction, KL, and total ELBO per epoch.

2) VAE Reconstruction & Generation (10 marks)

Show 5 validation images with reconstructions, side-by-side.
Sample 10 images by $z \sim \mathcal{N}(0, I_{16})$ and decode; show a 2×5 grid.
Add a brief comment on sharpness/diversity.

3) GAN (MLP-based) – Training (10 marks)

Latent: $d_z = 64$.
Generator G (use LeakyReLU(0.2) in hidden layers; last layer Tanh):
Linear(64→256) → LeakyReLU; Linear(256→512) → LeakyReLU; Linear(512→784) → Tanh.
Discriminator D (use LeakyReLU(0.2); Dropout 0.3 after hidden layers; output is a logit):
Linear(784→512) → LeakyReLU → Dropout(0.3); Linear(512→256) → LeakyReLU → Dropout(0.3); Linear(256→1).
Loss: BCEWithLogitsLoss, non-saturating objective. Label smoothing: real=0.9, fake=0.0.
Optimization: Adam(lr=2e-4, betas=(0.5,0.999)); batch=128; epochs=10; update D once and G once per iteration.
Output: Plot L_D and L_G per epoch.

4) GAN – Generation (10 marks)

Use a fixed batch of 64 noise vectors. Save 8×8 grids at epochs 1, 5, and 10.
Comment on progression or mode collapse.

5) Quantitative Comparison (20 marks)

Compute Inception Score (IS) and Fréchet Distance (FID) for 100 generated samples for both VAE and GAN using a fixed MNIST classifier you train:

Classifier (for IS/FID features):
Conv2d(1→32, k=3, s=1, p=1) → ReLU → MaxPool(2);
Conv2d(32→64, k=3, s=1, p=1) → ReLU → MaxPool(2);
Flatten → Linear(7·7·64→128) → ReLU → Linear(128→10) → Softmax.
Train 5 epochs on MNIST train (60k), Adam(lr=1e-3), batch=128; freeze afterward.

IS: Split 100 samples into 10 splits; IS = $\exp(\mathbb{E}[\mathrm{KL}(p(y|x) \,\|\, p(y))])$; report mean ± std across splits.
FID: Use 128-dim penultimate activations as features. Compute (μ_r, Σ_r) on 1,000 real test

images and (μ_g, Σ_g) on 100 generated; FID $= \|\mu_r - \mu_g\|_2^2 + \mathrm{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2})$.
Output: Table with IS (higher is better) and FID (lower is better) for VAE vs. GAN.

Part C – LLMs (Total: 40 marks)

Goal: Build and analyze a tiny autoregressive Transformer; compare decoding strategies; also report n-gram baseline perplexities.

Settings (Part C)

- Dataset: Tiny Shakespeare (character-level tokenization). Include newline as a token.
- Splits: Train 90% / Val 5% / Test 5% (contiguous split).
- Sequence length (context): 128; Batch size: 64; Epochs: 3.
- Optimizer: AdamW(lr=3e-4, weight_decay=0.01, betas=(0.9,0.95)); grad clip=1.0.
- Loss: Cross-entropy for next-token prediction. Initialize linear layers with Xavier uniform.

1) Tiny Transformer LM (10 marks)

Embedding: vocab size V (from data), embedding dim = 192.
Positions: sinusoidal (no learned positions).
Blocks: 2 identical Transformer blocks; each block contains:

1) Multi-Head Self-Attention: heads=3, head dim=64 (so $d\_model = 192$). Q/K/V linear projections: 192→192; output projection 192→192; causal mask over context length 128.
2) Residual + LayerNorm (pre-norm).
3) Feed-Forward: Linear(192→768) → GELU → Linear(768→192).
4) Residual + LayerNorm (pre-norm).

LM Head: Linear(192→ V).
Training: batch 64, seq len 128, epochs 3, AdamW as above; shuffle training batches each epoch.
Output: Plot train and validation loss per epoch.

2) Transformer Perplexity (10 marks)

Compute validation perplexity at the end of each epoch (teacher forcing). Report final Validation Perplexity (PPL). **Teacher forcing** is a training strategy used in autoregressive sequence models where the model is given the *ground-truth* token from the previous timestep as input instead of its own predicted token.

3) Decoding Strategies (20 marks)

Use the final Transformer to generate from the same prompt: the first 64 characters of the validation split.
Generate 200 tokens at temperature 1.0 for each strategy:

1. Greedy (argmax)
2. Top-k with $k = 20$

3. Nucleus (top-p) with $p = 0.9$

Outputs: Show all three generations (truncate in the notebook as needed). Comment on fluency vs. diversity.
