STAT 940 - Deep Learning

Assignment 2: Attention, Transformers, Alignment and VAEs

Fall 2026
Posted: October 26, 2026
Due: November 16, 2026, at 11:59 p.m. (Waterloo/Toronto time)
Weight: 15% of the course grade; 100 marks across six questions
Balance: 40 marks for conceptual/mathematical questions; 60 marks for computational work and analysis
Expected effort: approximately 17–22 hours over three weeks, including reading, debugging, analysis and submission. A modest CPU is sufficient; no GPU, paid API, external dataset or pretrained model is required.

Work and submission

This is an individual assignment. Show your reasoning and intermediate work.

Submit written answers through **Crowdmark**, under the matching question numbers. Include the requested tables, figures, formulas and interpretations. Do not paste full source code into Crowdmark.

Submit **one combined ZIP file** to **Learn → Assignment 2 Dropbox**. Name it `WatIAM_A2.zip`, replacing `WatIAM` with your username; for example, `jsmith_A2.zip`. This single ZIP must contain **both Q3 and Q6**, not one ZIP per question. Include your code, helpers, generated CSV tables and figures, and a **student-written** `README.md` with run commands, package versions and seeds. Use relative paths. Exclude environments, installed packages and trained-model checkpoints.

Use this layout (equivalent executable code filenames are acceptable):

```
README.md
Q3/q3.py
Q3/results/...
Q6/data.py
Q6/model.py
Q6/run_q6.py
Q6/check_loss.py
Q6/results/...
```

Use Python with NumPy and Matplotlib for Q3; Q6 also uses PyTorch. The Q6 starter supplies data generation, training, evaluation and plotting. Complete its model/sampling/loss functions and write `check_loss.py`. Equivalent independent code is permitted if it follows the protocol. A notebook must run from start to finish without manual intervention, with its execution command documented.

Late work: submissions up to 24 hours late receive a penalty of 10% of the earned mark; later submissions receive zero.

Use Piazza for assignment questions: public posts for general clarifications and private posts for personal matters.

Notation and sources

Use natural logarithms. Keep full precision in code; normally round displayed answers to four decimal places. Tokens are columns. With n_q query tokens and n_k key/value tokens,

$$Q \in \mathbb{R}^{p \times n_q}, \qquad K \in \mathbb{R}^{p \times n_k}, \qquad V \in \mathbb{R}^{m \times n_k}.$$

As in the lectures, p is the query/key width and m the value width. Define

$$S = K^T Q / \sqrt{p}, \qquad A_{ij} = \frac{e^{S_{ij}}}{\sum_{a=1}^{n_k} e^{S_{aj}}}, \qquad Y = VA.$$

Rows index keys and columns index queries; each column of A is a distribution over keys. This explicitly fixes the transpose and softmax axis for unequal sequence lengths. Transpose alternative source conventions consistently. Position-independent projection matrices are applied at every token position; query, key and value projections need not equal one another.

Superscript T denotes transpose; I_d is the d-dimensional identity; $\mathbf{1}_n$ is an n-vector of ones. The Frobenius norm $\|B\|_F$ is the square root of the sum of squared entries. The operation $\mathrm{diag}(v)$ places v on a matrix diagonal. A Gaussian's second scalar parameter is its **variance**, not its standard deviation. KL denotes Kullback–Leibler divergence; ELBO denotes the evidence lower bound on the **log** marginal likelihood. VAE means variational autoencoder.

Textbook references are to B. Ghojogh and A. Ghodsi, *Elements of Deep Learning*, Springer, 2026.

Question 1 — Attention and information flow (10 marks)

Adapted from Chapter 9, Exercise 5.

(a) Cross-attention, 3 marks. An encoder produces $H \in \mathbb{R}^{D_s \times n_s}$, and a decoder supplies $G \in \mathbb{R}^{D_t \times n_t}$. Use

$$Q = W_Q^T G, \qquad K = W_K^T H, \qquad V = W_V^T H.$$

1. Give the dimensions of W_Q, W_K, W_V, S, A, Y in a table and state which axis softmax normalizes.
2. For image captioning, let the columns of H represent image regions. In a short paragraph, identify the query, key and value roles and explain why n_s and n_t may differ.

(b) A numerical attention calculation, 4 marks. Use

$$q = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}, \quad K = \begin{bmatrix} 0.8 & 0.5 & 0.1 \\ 0.6 & 0.2 & 0.9 \end{bmatrix}, \quad V = \begin{bmatrix} 2 & 1 & 0 \\ 3 & 1 & 4 \end{bmatrix}.$$

Calculate and report the scaled-score vector, attention-weight vector and two-dimensional output. Show the substitutions and weighted-value calculation. You may use a calculator or short script.

(c) Causal masking, 3 marks. For n tokens, query j may attend only to keys $i \leq j$. Both pipelines below use causally masked attention, residual connections, positionwise feed-forward networks and layer normalization separately at each token. Here $e(x_j)$ is the token embedding and p_j is a fixed positional vector:

$$\text{A: } h_j = e(x_j) + p_j,$$

$$\text{B: } h_j = e(x_j) + p_j + \frac{1}{n}\sum_{t=1}^{n} e(x_t).$$

1. Write the additive score mask M_{ij}.
2. State which pipeline guarantees that changing a future token cannot affect output j. Justify your answer by tracing the input dependencies.
3. Give one minimal change that repairs the other pipeline.

Deliverables for Question 1

Crowdmark Q1: Dimensions and captioning roles; numerical scores, weights and demonstrated output calculation; causal mask, diagnosis and repair. No separate Q1 code upload is required.

Question 2 — Position and pretraining targets (10 marks)

Adapted from Chapter 9, Exercise 7; Chapter 11, Exercises 1 and 2.

(a) Test a claim about positional scores, 4 marks. Use positional vectors without token content:

$$p(t) = \begin{bmatrix} \sin t \\ \cos t \end{bmatrix}, \qquad p(t+k) = R(k)p(t), \qquad R(k) = \begin{bmatrix} \cos k & \sin k \\ -\sin k & \cos k \end{bmatrix}.$$

You may use the displayed shift identity and $R(k)^T R(k) = I_2$ without proof. A colleague concludes that, for arbitrary position-independent learned projections, the score

$$s(i,j) = \frac{(W_K^T p(i))^T (W_Q^T p(j))}{\sqrt{2}}$$

must depend only on the relative displacement $i - j$. Test this claim with

$$W_Q = I_2, \qquad W_K = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}.$$

1. Derive $s(i,j)$ for the supplied projections. Evaluate $s(0,0)$ and $s(1,1)$, then state whether they refute the claim.
2. Derive the score when $W_Q = W_K = I_2$. Explain why the shift identity gives a relative-only score in this case but not for arbitrary projections. You may use trigonometric identities without proof.

(b) Architecture and training targets, 6 marks. Submit two tables, each with one row for BERT, GPT and T5.

1. **Architecture table:** use columns **Model**, **Architecture**, **Self-attention context**, **Cross-attention source**, and **Main pretraining objective**. Use the original BERT-style encoder, GPT-style decoder-only model and T5-style encoder–decoder. For T5, distinguish encoder and decoder context. Omit auxiliary objectives.
2. **Input/target table:** use columns **Model**, **Input**, and **Target**, based on `the small cat sat on the mat`. Treat words as tokens. For BERT, replace one word with `[MASK]`; the target is that word. For GPT, choose a nonempty prefix that leaves a next word; that word is the target. For T5, remove two nonempty, nonoverlapping spans and replace them by `<extra_id_0>` and `<extra_id_1>` in order. These **sentinels** are special placeholder tokens. Write the target as each sentinel followed by its removed span, ending with `<extra_id_2>`.

Deliverables for Question 2

Crowdmark Q2: Score counterexample and explanation; architecture/context/objective table and three concrete input/target examples. No separate Q2 code upload is required.

Question 3 — Implement, audit and measure attention (30 marks)

Source: Chapter 9, Section 4.2; [R1].

For $x \in \mathbb{R}^p$, define

$$\phi_W(x)_a = r^{-1/2} \exp(w_a^T x - \|x\|^2/2), \qquad w_a \overset{\text{iid}}{\sim} \mathcal{N}(0, I_p), \quad a = 1, \dots, r.$$

Here r is the feature count and $W \in \mathbb{R}^{r \times p}$ has rows w_a^T. Use the **same** W for queries and keys. You may use

$$\mathbb{E}[\phi_W(x)^T \phi_W(y)] = e^{x^T y}.$$

Define $\Phi_Q[:, j] = \phi_W(p^{-1/4} q_j)$ and $\Phi_K[:, i] = \phi_W(p^{-1/4} k_i)$. Dense feature attention forms $B = \Phi_K^T \Phi_Q$, normalizes its columns and returns the weighted values. The equivalent associative form is

$$C = V \Phi_K^T \ (m \times r), \qquad u = \Phi_K \mathbf{1}_{n_k} \ (r),$$

$$N = C \Phi_Q \ (m \times n_q), \qquad h = u^T \Phi_Q \ (n_q),$$

$$\widehat{Y}_{aj} = N_{aj} / h_j.$$

Use these identities without proof and NumPy `float64` throughout. Exact attention must use stable columnwise softmax: subtract each column's maximum score before exponentiating. Feature attention is **unmasked**; only exact attention needs a causal-mask option. No training is required.

(a) Implement and test three paths, 12 marks.

Setup. Use `rtol=1e-10, atol=1e-10` for all array-agreement checks in (a) and (b). For the feature checks, draw these arrays in order and share them between both feature functions:

```
rng = np.random.default_rng(7)
Q = 0.5*rng.standard_normal((16,7))
K = 0.5*rng.standard_normal((16,11))
V = rng.standard_normal((8,11))
W = rng.standard_normal((64,16))
```

Implement. Write exact, dense feature and associative feature attention functions. Each feature function computes its own features. Return the output by default; an optional test flag may return intermediate arrays. The associative function must not form an $n_k \times n_q$ score, kernel or attention matrix. Do not clip denominators.

Check. Run these four assertions:

1. Assert that exact attention on Q1(b) returns `(1.1353806515790752, 2.6947119476120998)` as a `float64` array of shape `(2,1)`.
2. Assert that dense feature weights sum to one in each column.
3. Assert that both feature functions' normalizers are finite and strictly positive.
4. Assert that dense and associative feature outputs agree with each other.

Submit. Report all four assertion outcomes and the maximum absolute dense-versus-associative output difference.

Question 3, continued

(b) Test three attention properties, 6 marks.

Setup. Without positional information or dropout, unmasked self-attention with position-independent projections satisfies $F(XP) = F(X)P$: permuting input columns permutes output columns in the same way. Use this fact without proof. Draw these arrays in order:

```
rng = np.random.default_rng(31)
X = 0.5*rng.standard_normal((4,5))
UQ = rng.standard_normal((4,3))
UK = rng.standard_normal((4,3))
UV = rng.standard_normal((4,2))
perm = [2,0,4,1,3]
t = np.arange(5, dtype=np.float64)
E = np.vstack([np.sin(t), np.cos(t), np.sin(0.1*t), np.cos(0.1*t)])
```

Here $F(H)$ applies your exact routine to $Q = U_Q^T H, K = U_K^T H, V = U_V^T H$. Its default is unmasked; use the same matrices for all three checks.

Check.

1. Assert agreement between `F(X[:,perm])` and `F(X)[:,perm]`.
2. Compare `F(X[:,perm] + E)` and `F(X + E)[:,perm]`. Keep `E` fixed in the first expression; no agreement assertion is required.
3. Copy `X` and add `1.0` to every entry of its final column. Apply the causal mask from Q1(c) and assert agreement of the first four output columns before and after this change. Compute the maximum difference over those four columns only.

Submit. Give one three-row table with columns **Test**, **Maximum absolute difference**, **Expected behavior**, and **Assertion result**; enter "not required" for test 2's assertion. Then explain, in one sentence each, (i) why test 2 need not preserve the permutation identity and (ii) one masking error test 3 can detect.

Question 3, continued

(c) A controlled approximation and runtime study, 12 marks.

Setup. For each sequence length n, draw these arrays in order from a fresh generator:

```
rng = np.random.default_rng(2026+n)
GQ = rng.standard_normal((16,n))
GK = rng.standard_normal((16,n))
V = rng.standard_normal((8,n))
```

For each feature seed $s \in \{0, 1, 2\}$, draw:

```
W_full = np.random.default_rng(s).standard_normal((256,16))
```

Use its first r rows with $r^{-1/2}$ normalization. Reuse the same G_Q, G_K, V and feature matrices across input scales. Measure

$$\text{relative error} = \frac{\|\widehat{Y} - Y\|_F}{\|Y\|_F}, \qquad \text{mean column entropy} = -\frac{1}{n}\sum_j \sum_i A_{ij} \log A_{ij}, \quad 0 \log 0 = 0.$$

Run.

1. Compute relative errors at $n = 256$, $Q = 0.5G_Q, K = 0.5G_K$, for every $s \in \{0, 1, 2\}$ and $r \in \{16, 64, 256\}$: **nine measurements**.
2. Repeat at $Q = G_Q, K = G_K$, $n = 256, r = 256$, for the same three seeds: **three measurements**.
3. Compute exact-attention entropy at $n = 256$ for both input scales.
4. Time exact and associative attention at $n \in \{256, 512, 1024\}$, scale 0.5, using:
   `W = np.random.default_rng(0).standard_normal((128,16))`
   Include feature evaluation; exclude data/W generation and checks. At **each length**, warm up **each function** once; report the median of three calls timed with `time.perf_counter`. Before starting Python, set `OPENBLAS_NUM_THREADS=1`, `MKL_NUM_THREADS=1` and `OMP_NUM_THREADS=1` if feasible.

Submit.

1. A four-row error table: rows $(\text{scale}, r) = (0.5, 16), (0.5, 64), (0.5, 256), (1.0, 256)$; columns **Scale**, **r**, **Seed 0 error**, **Seed 1 error**, **Seed 2 error**, **Mean**.
2. A two-row table of input scale and exact-attention entropy; a three-row timing table with columns **n**, **Exact median time**, **Associative median time**. State the time unit, actual thread setting, CPU/platform and NumPy version.
3. Three numbered paragraphs, **at most 450 words total**: (i) feature-count and seed effects; (ii) the paired scale/entropy/error comparison; (iii) observed runtimes. Cite your results. Exclude tables, equations and environment details from the cap; no minimum length, monotonic error reduction or speedup is required.

Deliverables for Question 3

Crowdmark Q3: (a) Assertion outcomes and output difference; (b) three-row test table and two explanations; (c) error, entropy and timing tables, computing environment, and three interpretation paragraphs (at most 450 words total).
Learn: Put Q3 code, CSV results and any figures in `Q3/`. Include that folder **together with Q6 and your student-written** `README.md` **in one combined** `WatIAM_A2.zip`. Upload that one ZIP to the Assignment 2 Dropbox. Do not submit a separate Q3 ZIP or paste full code into Crowdmark.

Question 4 — What does preference optimization improve? (12 marks)

Source: Chapter 11, Sections 8–9; [R2].

(a) Diagnose three training errors, 3 marks. Check each rule and give the minimal correction.

A — Supervised fine-tuning (SFT). For a demonstration (x, y^*), let π_θ be the policy and ψ an independent parameter vector. Consider

$$L(\theta) = -\log \pi_\theta(y^* \mid x), \qquad \psi \leftarrow \psi - \eta \nabla_\psi L, \qquad \eta > 0.$$

Check: compute $\nabla_\psi L$ and correct the update.

B — Reward-model training. For preferred and dispreferred responses with scores r_w, r_l, the proposed pairwise loss is

$$L_{\text{RM}} = \log(1 + \exp[-(r_w + r_l)]).$$

Check: does swapping r_w and r_l change this loss? Correct it so minimizing the loss favors $r_w > r_l$.

C — Reinforcement learning from human feedback (RLHF). For a fixed prompt, frozen reward model $r(y)$ and frozen reference policy π_{ref}, **maximize**

$$J = \mathbb{E}_{y \sim \pi_\theta}[r(y)] + \beta \, \text{KL}(\pi_\theta \| \pi_{\text{ref}}), \qquad \beta > 0.$$

Two policies have equal expected reward but KL values 0.1 and 0.4. **Check:** which does this objective prefer? Correct it so the KL term discourages departure from the reference.

Submit **one three-row table (A, B, C)** with columns **Case**, **Check result**, and **Correction**. Each row: 0.5 mark for the check and 0.5 for the correction.

(b) Direct preference optimization (DPO) gradient, 5 marks. For prompt x, a softmax policy has logits (real-valued scores) u_w, u_l, u_o for three **complete responses**. The preference pair ranks y_w above y_l; y_o is absent. The fixed reference probabilities are $(0.40, 0.30, 0.30)$. For $\beta > 0$, define

$$\Delta(u) = \log \frac{\pi_u(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \log \frac{\pi_u(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)},$$

$$\ell(u) = -\log \sigma(\beta \Delta(u)), \qquad \sigma(a) = \frac{1}{1 + e^{-a}}.$$

For numerical evaluation, use

$$u = (\log 0.40, \log 0.30, \log 0.30), \qquad \beta = 1.$$

1. Derive $\partial \ell / \partial \Delta$ and $\nabla_u \ell$ for general u, β.
2. Calculate the loss and all three gradient components at the supplied point.
3. Explain why $\partial \ell / \partial u_o = 0$ does not imply that $\pi_u(y_o \mid x)$ stays fixed when the other logits change.

(c) Relative versus absolute improvement, 4 marks. Use the same reference and $\beta = 1$ to compare $\pi_A = (0.40, 0.30, 0.30)$ and $\pi_B = (0.35, 0.15, 0.50)$. Submit a two-row table of **Policy**, **Preferred/dispreferred probability ratio**, Δ, and **DPO loss**. Then answer, with justification: (i) Does lower loss imply higher absolute probability of the preferred response? (ii) Can this example establish improved overall quality without a quality label for y_o? The candidates need not be related by a gradient step.

Deliverables for Question 4

Crowdmark Q4: Three-row diagnosis table; symbolic and numerical gradient; two-policy calculation and interpretation. No separate Q4 code upload is required.

Question 5 — Audit a Gaussian variational calculation (8 marks)

Adapted from Chapter 12, Exercise 3.

Use

$$p(z) = \mathcal{N}(0,1), \quad p(x \mid z) = \mathcal{N}(z,1), \quad q(z) = \mathcal{N}(\mu, s^2), \quad s > 0, \quad x = 1.5.$$

Initially $\mu = 0.2, s = 0.5$; s is a **standard deviation**. This model has exact posterior $p(z \mid x) = \mathcal{N}(x/2, 1/2)$ and marginal $p(x) = \mathcal{N}(0,2)$. You may use these facts and the following identities without proof:

$$\mathcal{L} = \mathbb{E}_q[\log p(x \mid z)] - \mathrm{KL}(q\|p), \qquad \log p(x) - \mathcal{L} = \mathrm{KL}(q\|p(z \mid x)),$$

$$\mathrm{KL}(\mathcal{N}(a,v)\|\mathcal{N}(b,w)) = \frac{1}{2}\left[\frac{v + (a-b)^2}{w} - 1 + \log\frac{w}{v}\right], \qquad \mathbb{E}_q[(x-z)^2] = (x-\mu)^2 + s^2.$$

(a) Numerical consistency, 4 marks. At the initial q, calculate the following, retaining all Gaussian normalizing constants:

1. Expected log-likelihood, KL to the prior, and ELBO.
2. $\log p(1.5)$, the ELBO gap, and KL to the exact posterior. Check that the last two agree.

Report these six values with substitutions, then state the (μ, s^2) that makes the bound tight. You may use a short script.

(b) Diagnose the update, 4 marks. A colleague proposes subtracting the ELBO gradient to maximize $\mathcal{L}$, and claims that a tight bound requires $q(z) = p(z)$.

1. Derive $\partial\mathcal{L}/\partial\mu$ and $\partial\mathcal{L}/\partial s$.
2. Correct the update sign and take one **simultaneous** step of size 0.1 from the initial parameters. Report the new mean, standard deviation, ELBO, and signed ELBO change.
3. Use the exact posterior to explain the error in the tightness claim.

Deliverables for Question 5

Crowdmark Q5: Numerical ELBO and gap check, including substitutions and tight parameters; gradients, corrected update, ELBO change and diagnosis. No separate Q5 code upload is required.

Question 6 — A small VAE and a reconstruction-only control (30 marks)

Source: Chapter 12, Section 3; [R3].

Task. Implement and check the model and loss in (a), run the four prescribed experiments in (b), and interpret their results in (c). The starter supplies data generation, training, evaluation and plotting.

Setup — Model and loss

Use latent $z \in \mathbb{R}^2$, prior $p(z) = \mathcal{N}(0, I_2)$, and

$$q_\phi(z \mid x) = \mathcal{N}(\mu_\phi(x), \mathrm{diag}(e^{\ell_\phi(x)})), \qquad p_\theta(x \mid z) = \mathcal{N}(g_\theta(z), \tau^2 I_2), \quad \tau = 0.25.$$

Here ℓ is log-variance and τ is fixed. Per example, minimize

$$L_\lambda(x) = \mathbb{E}_q\left[\frac{\|x - g_\theta(z)\|^2}{2\tau^2} + \log(2\pi\tau^2)\right] + \lambda\frac{1}{2}\sum_{j=1}^{2}\left(\mu_j^2 + e^{\ell_j} - 1 - \ell_j\right).$$

- Use one reparameterized draw per training example; sum over coordinates and average over examples.
- Omit the fixed $\log(2\pi\tau^2)$ constant in code and training curves.
- Compare $\lambda \in \{0, 1\}$ at seeds 11 and 22: **four runs total**.

Data — supplied in `data.py`

```
C = np.array([[-2,-2],[-2,2],[2,-2],[2,2]], dtype=float)
rng = np.random.default_rng(data_seed)
labels = rng.integers(0,4,size=N)
X = C[labels] + 0.20*rng.standard_normal((N,2))
```

- Training: 2,048 observations, seed 2026. Held-out: 512 observations, seed 2027.
- Use PyTorch `float32`, without standardization; labels are for plotting only.

Architecture — implement in `model.py` (or equivalent code)

- Encoder: shared linear layers `2 → 32 → 32`, each followed by ReLU; separate linear `32 → 2` heads for μ and raw ℓ. Clamp ℓ to $[-8, 8]$ before sampling and KL evaluation.
- Decoder: `2 → 32 → 32 → 2`, ReLU after hidden layers, linear output.
- Use biases and default PyTorch linear-layer initialization.

Training protocol — supplied in `run_q6.py`

- CPU, one Torch thread, 100 epochs, batch size 128; Adam learning rate 10^{-3}, other defaults. No weight decay, scheduler or early stopping; do not tune the architecture or training budget.
- Reset `torch.manual_seed(s)` for each fresh model. Use separate CPU generators: $s + 1000$ for epoch shuffling (`torch.randperm`), $s + 2000$ for training latent noise.
- At each seed, both λ's must share initialization, batch orders and noise; keep model construction order fixed.

Question 6, continued

(a) Implement and check the model and loss, 12 marks. Implement the specified model, sampler and loss. In the starter, complete the five TODOs in `model.py`: `VAE.__init__`, `VAE.encode`, `sample_latent`, `analytic_kl`, and `loss_terms`. The loss function must return reconstruction and KL vectors of shape `(batch,)`; the training loop applies λ and averages them. Use the dedicated sampling generator. If using the starter, retain its function signatures.

1. **Run the sampler/KL diagnostic.** The supplied check uses your sampler with $\mu = (0.5, -0.5)$, $\ell = (\log 0.25, \log 4)$, 50,000 draws, and CPU-generator seed 31415. Required checks: means within 0.05 of target; population variances within 5% of $(0.25, 4)$; per-example KL equal to **1.375** with `rtol=atol=1e-6`; finite losses and KL $\geq -10^{-6}$. Independent implementations must reproduce these checks. Report the reparameterization formula, observed means, variances, KL and pass/fail results. Explain which log-variance/standard-deviation error the sampling test detects.
2. **Write `Q6/check_loss.py`.** Import your actual loss function (`loss_terms` in the starter). Use `X=[[1,-1],[0,2]]`, zero reconstructed outputs, and $\mu = \ell = 0$, all of shape `(2,2)`. Hand-calculate the two reconstruction terms without the constant, the two KL terms, and mean loss for $\lambda = 1$. Assert the returned vector shapes, values and resulting mean using `rtol=atol=1e-6`. Report expected and observed values together and explain how the test detects incorrect coordinate averaging or premature batch reduction. Put its run command in your README.

(b) Run and report the experiment, 8 marks. Run the four models under the fixed protocol. The supplied `run_q6.py` handles evaluation and plotting. Evaluate held-out data without gradients. MSE uses $g_\theta(\mu_\phi(x))$, averaged over examples and both coordinates. Mean KL sums latent-coordinate contributions before averaging over examples.

For generation, the script resets a separate CPU generator to $s+3000$, draws 2,048 standard-normal latent vectors, then 2,048 independent two-dimensional standard-normal noise vectors, and forms

$$\tilde{x} = g_\theta(z) + \tau\xi.$$

Reuse these draws for both λ's at each seed. "Within radius" means distance at most 0.6 from at least one listed center; nearest-center ties use center order.

Submit these outputs from the four runs:

1. A four-row table with columns **Seed**, λ, **Held-out MSE**, **Mean KL**, **Within-radius fraction**, **Count outside** $[-4, 4]^2$.
2. A second four-row table with **Seed**, λ and the four **Nearest-center fractions**, in the listed center order.
3. Training curves of epoch-mean reconstruction (without its constant) and **unweighted** KL, with all four runs labeled; one held-out scatter plot; and a 2-by-2 grid of generated-sample plots. All scatter plots must mark centers and use equal aspect ratio and $[-4, 4]$ on both axes. Label any log scale used for KL.

Question 6, continued

(c) Interpret the results, 10 marks. For a decoder mean exactly at one center, Gaussian observation noise with standard deviation 0.25 gives probability

$$1 - \exp[-0.6^2/(2 \times 0.25^2)] \approx 0.9439.$$

of landing within 0.6 of **that center**. Use this as a noise reference, not a strict ceiling for the fraction near any center or a finite sample.

Submit a two-row table (one row per seed) of signed changes, $\lambda = 1$ minus $\lambda = 0$, in **Held-out MSE** and **Within-radius fraction**. Then answer these four prompts in numbered paragraphs, **at most 500 words total**, excluding tables and equations; there is no minimum:

1. What reconstruction/generation tradeoff do the changes show? Is the conclusion consistent across both seeds? Cite the signed changes.
2. Why can accurate reconstruction coexist with poor samples drawn from the prior?
3. What is one limitation of the nearest-center fractions? Relate it to a reported value or scatter plot.
4. Use the noise reference to interpret your within-radius fractions. Identify one specific surprise or remaining limitation supported by a reported value or plot.

Use only the prescribed runs. Conclusions must follow your results; no particular ranking is required.

Deliverables for Question 6

Crowdmark Q6: (a) Sampling formula, diagnostic results, expected/observed loss-check values and explanations; (b) two four-row metric tables and specified plots; (c) two-row change table and four interpretation paragraphs (at most 500 words total).
Learn: Put completed `model.py`, supplied `data.py` and `run_q6.py`, student-written `check_loss.py` (or equivalent executable code), CSV results and figures in `Q6/`. Include this folder **in the same single** `WatIAM_A2.zip` **as Q3**, with your student-written `README.md`. Upload one combined ZIP to the Assignment 2 Dropbox, not separate Q3 and Q6 ZIPs.

Supplemental references

[R1] K. Choromanski et al., *Rethinking Attention with Performers*, ICLR 2021, Section 2.3, Lemma 1. arXiv:2009.14794.

[R2] R. Rafailov et al., *Direct Preference Optimization: Your Language Model Is Secretly a Reward Model*, NeurIPS 2023. arXiv:2305.18290.

[R3] D. P. Kingma and M. Welling, *Auto-Encoding Variational Bayes*, ICLR 2014. arXiv:1312.6114.