Deep Learning

Assignment 1

Foundations, Optimization, Regularization, and Neural Architectures

Due: Monday, October 26, 2026, 11:59 p.m. **100 marks**

Time zone: America/Toronto (Eastern Time).

Submission. This is an individual assignment. Upload written solutions, tables, and figures to matching questions in **Crowdmark**. For Learn, **combine all required code and supporting files for BOTH Q1 and Q7 into ONE ZIP**. Name it `WatIAM_A1.zip`, replacing `WatIAM` with your Waterloo username (e.g., `jsmith_A1.zip`). **After completing both questions, upload this single combined ZIP once to the Assignment 1 Dropbox in Learn.** Do not submit separate Q1 and Q7 ZIPs or individual code files.

Include `Q1.py` (or `Q1.R`/`Q1.m`), `Q7.py`, `data.py`, `model.py`, helpers, the three CSV files listed in Q7's box, and **one README covering both questions** (commands, dependencies, and seeds). Do not paste full code into Crowdmark.

Programming. For Q1, implement your analytic gradients and finite-difference check using basic arrays; do not use automatic differentiation. For Q7, use Python/NumPy and the supplied engine. Write your own experiment loop, comparisons, logging, selection, and diagnostics; do not call ready-made training, search, or early-stopping routines. Use relative paths so code runs from the extracted ZIP without editing.

Notation and scope. Matrices are uppercase, vectors are lowercase bold, and scalars are lowercase. Observations are columns:

$$X = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d \times n},$$

where n counts observations and d features. Superscript $\top$ means transpose; $\|\mathbf{w}\|$ is the Euclidean norm. Symbols are local to each question; logarithms are natural. Allow **16–22 hours over three calendar weeks**, including reading, debugging, writing, and submission preparation. Reading Week is October 10–18, with no intermediate deadline. No GPU or dataset download is required. Show your reasoning and checks.

Generative AI survey (required). After completing the assignment, open Learn and choose **Submit** → **Surveys** → **Assignment 1: Generative AI Use Survey**. Complete it even if you used no generative AI (answer **No to all five questions**). AI use is permitted; your survey responses do not affect your assignment grade. The survey is named so completion can be checked. You remain responsible for understanding your submitted work.

Individual Verification of Understanding: You may be asked to complete a brief individual quiz or participate in an oral discussion about your assignment. This may include explaining your reasoning and solutions, answering related questions, or explaining and making small modifications to your submitted code. You must demonstrate your own understanding of all submitted work, including work developed with permitted generative AI assistance. These checks must be completed without generative AI or assistance from others. If you cannot demonstrate adequate understanding of your submitted work, your overall assignment mark may be reduced.

Late Submission Policy: Submissions received up to 24 hours after the deadline receive a 10% penalty. Submissions more than 24 hours late receive a grade of zero.

Question 1: Representation and a backpropagation audit

[16 marks]

Consider the four observations below. Each column of X is an input $\mathbf{x}_i$; the corresponding entry of $\mathbf{y}$ is its label:

$$X = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \qquad \mathbf{y} = (-1, +1, +1, -1)^\top.$$

These are exclusive-or (XOR) labels: the label is positive exactly when the two input coordinates differ.

(a) [6 marks] A **single threshold neuron**, or perceptron classifier, assigns a label by thresholding a weighted sum plus a bias:

$$f(\mathbf{x}) = \begin{cases} +1, & \mathbf{w}^\top \mathbf{x} + b \geq 0, \\ -1, & \mathbf{w}^\top \mathbf{x} + b < 0. \end{cases}$$

Here $\mathbf{x}, \mathbf{w} \in \mathbb{R}^2$, $b \in \mathbb{R}$, and $\mathbf{w}^\top \mathbf{x} = w_1 x_1 + w_2 x_2$. This is called an **affine threshold classifier**: "affine" means a linear expression plus a constant bias. To *separate* the observations means to assign all four labels correctly. Prove that no classifier of this form separates them.

Next construct a network with two hidden threshold units and one output. We use the **indicator notation**

$$1\{\text{condition}\} = \begin{cases} 1, & \text{if the condition is true,} \\ 0, & \text{if the condition is false.} \end{cases}$$

The required network has the form

$$\begin{aligned} h_j(\mathbf{x}) &= 1\{\mathbf{w}_j^\top \mathbf{x} + b_j \geq 0\}, \qquad\qquad j = 1, 2, \\ \mathbf{h}(\mathbf{x}) &= (h_1(\mathbf{x}), h_2(\mathbf{x}))^\top, \\ \widehat{y}(\mathbf{x}) &= 2 \cdot 1\{\mathbf{a}^\top \mathbf{h}(\mathbf{x}) + d \geq 0\} - 1. \end{aligned}$$

Each hidden unit outputs 1 when its weighted sum plus bias is nonnegative, and 0 otherwise. The vectors $\mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^2$ and biases b_1, b_2 belong to the hidden units; $\mathbf{a} \in \mathbb{R}^2$ and d are the output weights and bias. The output expression converts an indicator value of 0 to label -1, and 1 to label $+1$.

List all weights and biases and verify the four outputs in a table.

(b) [6 marks] Consider instead the smooth network

$$\mathbf{h}_i = \tanh(W\mathbf{x}_i + \mathbf{b}), \qquad \widehat{y}_i = \mathbf{v}^\top \mathbf{h}_i + c, \qquad \ell = \frac{1}{2n} \sum_{i=1}^{n} (\widehat{y}_i - y_i)^2.$$

Here $W \in \mathbb{R}^{2 \times 2}$, $\mathbf{b}, \mathbf{v} \in \mathbb{R}^2$, $c \in \mathbb{R}$, and tanh is applied separately to each coordinate. The four parameter blocks are $W, \mathbf{b}, \mathbf{v}, c$. Derive all four gradients and state their dimensions.

Prove or refute this claim: because two hidden units can represent XOR, gradient descent will learn XOR from the initialization

$$W = 0, \qquad \mathbf{b} = \mathbf{v} = \mathbf{0}, \qquad c = 0.$$

Use full-batch gradient descent, which uses all four observations in every update, with no momentum or random perturbation.

(c) [4 marks] Use the same observations, so $n = 4$. Check the directional derivative of the loss at

$$W = \begin{pmatrix} .3 & -.2 \\ .4 & .1 \end{pmatrix}, \qquad \mathbf{b} = (.1, -.1)^\top, \qquad \mathbf{v} = (.7, -.5)^\top, \qquad c = .2.$$

Arrange the parameters into one vector by reading the rows of W in order:

$$\boldsymbol{\theta} = (w_{11}, w_{12}, w_{21}, w_{22}, b_1, b_2, v_1, v_2, c)^\top, \qquad \mathbf{d} = \frac{(1, -2, 3, -4, 5, -6, 7, -8, 9)^\top}{\sqrt{285}}.$$

Here w_{jk} denotes entry (j, k) of W. The directional derivative measures how the loss changes along $\mathbf{d}$. Compare your derived analytic value with the centered finite difference:

$$\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta})^\top \mathbf{d} \quad \text{versus} \quad \frac{\ell(\boldsymbol{\theta} + s\mathbf{d}) - \ell(\boldsymbol{\theta} - s\mathbf{d})}{2s}, \qquad s \in \{10^{-2}, 10^{-4}, 10^{-6}\}.$$

Report the values in a table and explain their agreement. Explain why checking only at the zero initialization is a weak check. Implement your derived gradients and the finite differences using basic array operations; automatic differentiation is not permitted for this check.

Deliverables for Question 1

Crowdmark Q1: Parts (a)–(c): the nonseparability proof and checked threshold network; all gradient blocks and dimensions; the zero-initialization diagnosis; and the directional-derivative table with its interpretation.

Code / Learn: After completing **both Q1 and Q7**, put `Q1.py` (or `Q1.R`/`Q1.m`), its helpers, **all required Q7 files listed in the Q7 box**, and **one README with run commands for both questions** into **ONE combined ZIP**. Name it `WatIAM_A1.zip`, replacing `WatIAM` with your Waterloo username (for example, `jsmith_A1.zip`). Upload this single ZIP once to the **Assignment 1 Dropbox in Learn**. Do not submit separate ZIPs for Q1 and Q7 or upload code files separately. Do not paste full code into Crowdmark.

Question 2: Optimizer claims under controlled assumptions

[12 marks]

Consider the quadratic objective

$$q(\mathbf{w}) = \tfrac{1}{2}\mathbf{w}^\top A \mathbf{w}, \qquad A = \begin{pmatrix} 1 & 0 \\ 0 & 25 \end{pmatrix}, \qquad \mathbf{w} \in \mathbb{R}^2.$$

The symbol ∇q denotes its gradient; η is the learning rate.

(a) [4 marks] Derive the exact range of constant $\eta > 0$ for which gradient descent converges to zero from every starting point. For $\eta = .1$, give each coordinate after t updates in terms of its initial value. Identify which coordinate can diverge, its growth factor per update, and the initial condition that avoids this divergence.

For momentum use the following convention, where $\mathbf{v}_t$ is an accumulated gradient and β is the momentum coefficient:

$$\mathbf{v}_{t+1} = \beta \mathbf{v}_t + \nabla q(\mathbf{w}_t), \qquad \mathbf{w}_{t+1} = \mathbf{w}_t - \eta \mathbf{v}_{t+1}, \qquad \mathbf{v}_0 = \mathbf{0}.$$

If z_t is the scalar coordinate of $\mathbf{w}_t$ along an eigenvector of A with eigenvalue λ, you may use

$$z_{t+1} = (1 + \beta - \eta\lambda) z_t - \beta z_{t-1}.$$

Determine whether $\eta = .1, \beta = .8$ gives convergence. Use the *characteristic roots*, obtained by substituting $z_t = r^t$ into this recurrence; convergence requires both roots to have absolute value less than one.

(b) [4 marks] Let ℓ_i be the loss for observation i, with empirical mean loss

$$\ell(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} \ell_i(\mathbf{w}).$$

Write $\widehat{\mathbf{g}}(\mathbf{w})$ for the average gradient from a mini-batch of b observations sampled independently and uniformly with replacement. Use fresh samples at each iteration. You may use

$$\mathbb{E}[\widehat{\mathbf{g}}(\mathbf{w}) \mid \mathbf{w}] = \nabla \ell(\mathbf{w}), \qquad \mathrm{Cov}(\widehat{\mathbf{g}}(\mathbf{w}) \mid \mathbf{w}) = \frac{C(\mathbf{w})}{b},$$

where $C(\mathbf{w})$ is the covariance matrix of one sampled gradient. These expectations and covariances are over mini-batch sampling with the current iterate held fixed. Assume the required expectations and covariances are finite. No proof of these supplied identities is required.

Diagnose: "Unbiased stochastic gradients imply that stochastic gradient descent (SGD) and full-batch gradient descent (GD) have the same expected trajectory for every smooth objective." Compare methods with the same deterministic initial iterate and learning-rate sequence. Give the correct conditional one-step statement, explain why it does not establish the claim over several updates, and state one condition under which the expected trajectories agree.

(c) [4 marks] An exact Newton step solves the positive-definite quadratic above in one iteration. Does this make Newton's direction a descent direction for every smooth loss? Analyze

$$f(w_1, w_2) = \tfrac{1}{2}w_1^2 - \tfrac{1}{2}w_2^2 + \tfrac{1}{4}w_2^4 \quad \text{at} \quad (w_1, w_2) = (0, \tfrac{1}{2}).$$

Compute the directional derivative $\nabla f^\top \mathbf{d}$, where

$$\mathbf{d} = -[\nabla^2 f]^{-1} \nabla f.$$

Here $\nabla^2 f$ is the Hessian matrix of second derivatives, and both derivatives are evaluated at the stated point. A descent direction has negative directional derivative. State a sufficient Hessian condition for descent when the gradient is nonzero. Compute the loss before and after the full step $\mathbf{w} + \mathbf{d}$, and explain why this comparison need not agree with the local directional sign.

Deliverables for Question 2

Crowdmark Q2: Parts (a)–(c): learning-rate and characteristic-root calculations; conditional expectation and expected-trajectory critique; and the Newton-direction calculation and explanations.

Code / Learn: No code or separate Learn file is required for Question 2.

Question 3: Risk estimation and evidence for generalization

[14 marks]

At n fixed input locations, observe a response vector

$$\mathbf{y} = \boldsymbol{\mu} + \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 I_n).$$

Here $\boldsymbol{\mu} \in \mathbb{R}^n$ contains the unknown true mean responses, σ^2 is the known noise variance, and I_n is the $n \times n$ identity matrix. Expectations in this question are over the response noise, with the inputs and $\boldsymbol{\mu}$ held fixed.

A **fixed linear smoother** is a linear estimation rule

$$\widehat{\boldsymbol{\mu}} = H\mathbf{y}, \qquad H \in \mathbb{R}^{n \times n},$$

where H is deterministic and does not depend on the observed responses. No symmetry or projection assumption is imposed on H.

For an estimator $\mathbf{g}(\mathbf{y})$, its **mean-estimation risk** is

$$r(\mathbf{g}) = \frac{1}{n}\mathbb{E}\|\mathbf{g}(\mathbf{y}) - \boldsymbol{\mu}\|^2.$$

You may use **Stein's unbiased risk estimate (SURE)** without proof:

$$\widehat{r}(\mathbf{g}) = \frac{1}{n}\|\mathbf{y} - \mathbf{g}(\mathbf{y})\|^2 + \frac{2\sigma^2}{n}\sum_{i=1}^{n}\frac{\partial g_i}{\partial y_i} - \sigma^2, \qquad \mathbb{E}[\widehat{r}(\mathbf{g})] = r(\mathbf{g}).$$

The identity requires differentiability and integrability conditions. They hold for the fixed linear smoothers in part (a); you need not establish the general conditions.

(a) [4 marks] Specialize the supplied formula to $\mathbf{g}(\mathbf{y}) = H\mathbf{y}$, and explain why response-independence of H matters. State the adjustment for predicting a new noisy response vector at the same inputs, with independent noise of the same variance. Explain why neither risk by itself concerns a new input distribution. No proof of SURE is required.

(b) [6 marks] For $n = 1, \mu = 0, \sigma^2 = 1$, compare the two fixed scalar smoothers $h_0 = 0$ and $h_1 = 1$, which produce estimates $h_0 y$ and $h_1 y$. Select the one with the smaller estimated mean-estimation risk from part (a), and derive this selection rule.

Write the chosen coefficient as $\widehat{h}(y)$. Prove that the expectation of the reported minimum risk estimate is below the actual risk of the selected estimator:

$$r_{\text{sel}} = \mathbb{E}[(\widehat{h}(y)y - \mu)^2].$$

This expectation includes the randomness of both the selection and its resulting estimate. Hint: show that the **expected reported minimum is negative**, while the **actual selected-estimator risk is positive**. Compare these expectations, not the reported estimate and realized loss for each response. No Gaussian integral is required. Explain why the conclusion does not contradict fixed-smoother unbiasedness.

(c) [4 marks] For each statement, identify what the stated evidence establishes and one specific missing control or measurement. Keep the combined response under 250 words.

1. A width sweep (varying the number of hidden units), evaluated on one fixed validation set, shows zero training error above a threshold and a validation-error peak followed by a decline. "This proves every wider network generalizes better."
2. Training error is near zero early; validation accuracy rises sharply much later at a fixed width. "This is ordinary overfitting, so the later checkpoints must be worse."
3. Two interpolating networks (both fit their training observations exactly) have different test error although neither used an explicit penalty. "Regularization cannot explain the difference."

Use the lecture terms **double descent**, **grokking**, and **implicit regularization** carefully; a diagnostic curve alone does not establish its cause.

Deliverables for Question 3

Crowdmark Q3: Parts (a)–(c): the specialized SURE formula, assumptions and risk adjustment; the selection rule and bias proof; and the three evidence critiques (250 words combined).

Code / Learn: No code or separate Learn file is required for Question 3.

Question 4: What regularization and normalization actually change

[14 marks]

For linear regression with a scalar response, weights $\mathbf{w} \in \mathbb{R}^d$, and bias c, let

$$\ell(\mathbf{w}, c) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{w}^\top \mathbf{x}_i - c)^2.$$

The bias is never perturbed or penalized. Perturb input $\mathbf{x}_i$ by adding a noise vector $\boldsymbol{\xi}_i$, where

$$\mathbb{E}[\boldsymbol{\xi}_i] = \mathbf{0}, \qquad \mathrm{Cov}(\boldsymbol{\xi}_i) = \Sigma_i.$$

The covariance Σ_i may differ between observations. You may use

$$\mathbb{E}[\ell_{\text{noise}}(\mathbf{w}, c)] = \ell(\mathbf{w}, c) + \frac{1}{n}\sum_{i=1}^{n}\mathbf{w}^\top \Sigma_i \mathbf{w}.$$

Here ℓ_{noise} is the same mean squared loss with perturbed inputs. The expectation is over the injected noise, with the data fixed. No proof of this identity is required.

(a) [6 marks] In **inverted input dropout**, each retained feature is divided by its retention probability q:

$$\widetilde{x}_{ji} = \frac{m_{ji}x_{ji}}{q}, \qquad m_{ji} \sim \text{Bernoulli}(q), \qquad 0 < q \leq 1.$$

The index j denotes a feature and i an observation. All masks are independent; each mask is 1 with probability q and 0 otherwise.

Derive the expected dropout loss as the original loss plus a penalty. Determine when there is a constant $\rho \geq 0$, independent of $\mathbf{w}$, such that the penalty is

$$\rho\|\mathbf{w}\|^2 \qquad \text{for every } \mathbf{w}.$$

This is an **isotropic** penalty: it treats every weight direction equally. Treat $0 < q < 1$ and the zero-penalty case $q = 1$ separately. For $q < 1$, construct a two-feature dataset whose two positive diagonal penalty coefficients have a ratio of at least 100, and check that ratio.

For a penalty $\lambda\|\mathbf{w}\|^2$, $\lambda \geq 0$, added to the stated loss, write one plain-SGD update with learning rate $\eta > 0$ and identify its multiplicative weight factor. Keep the bias unpenalized.

(b) [4 marks] Now let every observation have the same input-noise covariance

$$\Sigma = \tau^2 \mathbf{t}\mathbf{t}^\top, \qquad \|\mathbf{t}\| = 1, \qquad \tau > 0.$$

Here $\mathbf{t}$ is a fixed direction in input space and τ sets the noise magnitude. Interpret the supplied penalty as discouraging changes in the prediction along $\mathbf{t}$. Give one concrete case in which that invariance damages prediction.

For k-class cross-entropy, label smoothing replaces a one-hot target by

$$\widetilde{\mathbf{y}} = (1 - \alpha)\mathbf{e}_j + \frac{\alpha}{k}\mathbf{1}_k, \qquad 0 < \alpha < 1.$$

Here j is the correct class, $\mathbf{e}_j \in \mathbb{R}^k$ has a 1 in position j and zeros elsewhere, and $\mathbf{1}_k \in \mathbb{R}^k$ is the all-ones vector. Bold $\mathbf{1}_k$ denotes a vector; plain $1\{\text{condition}\}$ in Q1 denotes a scalar indicator.

The **logits z** are the class scores before softmax, and **p** is the resulting probability vector. You may use

$$\nabla_{\mathbf{z}} \ell = \mathbf{p} - \widetilde{\mathbf{y}}.$$

Identify the probability vector that minimizes cross-entropy for one example when its logits can vary freely. Explain briefly why this target change and the directional noise penalty are not generally equivalent to an isotropic penalty $\lambda\|\mathbf{w}\|^2$. Do not derive the supplied logit gradient.

(c) [4 marks] For one feature, batch normalization (**BatchNorm**) acts on its activation values $\mathbf{a} \in \mathbb{R}^b$ across a batch of b examples:

$$\mu_b = \frac{1}{b}\sum_{i=1}^{b} a_i, \qquad s_b^2 = \frac{1}{b}\sum_{i=1}^{b}(a_i - \mu_b)^2, \qquad z_i = \gamma \frac{a_i - \mu_b}{\sqrt{s_b^2 + \varepsilon}} + \beta.$$

Here γ is a learned scale, β is a learned shift, and $\varepsilon > 0$ is a fixed numerical stabilizer. This β is unrelated to Q2's momentum coefficient. For a positive rescaling $r > 0$, you may use

$$\frac{r(a_i - \mu_b)}{\sqrt{r^2 s_b^2 + \varepsilon}} = \frac{a_i - \mu_b}{\sqrt{s_b^2 + \varepsilon/r^2}}.$$

Address the following points:

- Explain exact invariance to a common additive shift, and assess positive-rescaling invariance with positive or zero ε. For zero ε, assume positive batch variance; state what fails at zero variance.
- Name one normalization alternative whose output for an example is independent of the other examples. Identify its normalization axis: which entries supply its normalization statistics or norm?

Deliverables for Question 4

Crowdmark Q4: Parts (a)–(c): dropout derivation, isotropy condition and checked dataset; weight update; tangent and smoothing interpretations; and the BatchNorm shift/rescaling analysis with variance caveats and one normalization alternative with its axis.

Code / Learn: No code or separate Learn file is required for Question 4.

Question 5: Architecture and invariance audit

[10 marks]

A convolutional neural network (**CNN**) takes a 16×16 grayscale image with one input channel. Apply the following stages in order:

Stage	Operation
1	Valid 3×3 cross-correlation, four output channels, then ReLU.
2	2×2 max pooling with stride two.
3	Valid 3×3 cross-correlation, eight output channels, then ReLU.
4	2×2 max pooling with stride two.
5	Flatten to a vector, then a fully connected (dense) layer with three outputs.

"Valid" means no padding. Cross-correlation is the usual CNN convolution operation without flipping the kernel. Convolutional stride is one; each convolutional output channel and dense output has its own bias. Incomplete pooling windows are discarded. ReLU applies $\max(0, a)$ to each activation a.

(a) [4 marks] Give the height, width, and channel count after each spatial stage, the trainable parameter count of each layer and the total, and the receptive-field size of one final pooled activation. Here a **receptive field** is the region of the input image that can affect that activation. Explain how parameter sharing differs from an unshared locally connected layer.

(b) [3 marks] At the $7 \times 7 \times 4$ feature map (height $\times$ width $\times$ channels), a proposed residual block adds the output of a valid 3×3, eight-channel transformation to its input. Diagnose the mismatch. Give a residual redesign that retains spatial size and has eight output channels; specify padding on the transformation path and the shortcut that bypasses it.

For a DenseNet-style alternative, concatenate the original four channels with eight newly produced channels at the same spatial size. Give the resulting channel count and one memory or parameter implication for the following layer.

(c) [3 marks] Critique "Convolution followed by stride-two max pooling is invariant to every one-pixel translation." Use the **identity convolution**, which leaves the input unchanged. Provide a length-four one-dimensional input and its circular one-position shift (the entry at the boundary wraps to the other end) for which nonoverlapping width-two max pooling gives different outputs.

State the difference between equivariance and invariance, and name a boundary assumption needed for exact convolutional translation equivariance.

Deliverables for Question 5

Crowdmark Q5: Parts (a)–(c): the shape/parameter table and receptive field; residual redesign and dense-concatenation explanation; and a checked pooling counterexample with the requested definitions and boundary condition.

Code / Learn: No code or separate Learn file is required for Question 5.

Question 6: Recurrent gradients and the limits of gates

[12 marks]

A scalar recurrent neural network (**RNN**) processes inputs $x_1, \ldots, x_m$. Its hidden state is h_t, its target at the last step is y, and its initial state h_0 is fixed:

$$h_t = \tanh(wh_{t-1} + ux_t), \qquad t = 1, \ldots, m, \qquad \ell = \tfrac{1}{2}(h_m - y)^2.$$

The recurrent weight w and input weight u are shared across all time steps. Hold the input values fixed when differentiating. For the preactivation

$$a_t = wh_{t-1} + ux_t,$$

you may use the **backpropagation through time (BPTT)** recurrence

$$\delta_m = (h_m - y)(1 - h_m^2), \qquad \delta_t = w(1 - h_t^2)\delta_{t+1} \quad (t < m), \qquad \delta_t = \frac{\partial \ell}{\partial a_t}.$$

(a) [4 marks] Obtain the gradients with respect to w, u, and explain why each is a sum over time. Derive the **state sensitivity** $\partial h_m / \partial h_0$, which describes how the final state changes when the initial state is perturbed. For this sensitivity, vary h_0 while keeping inputs and weights fixed.

(b) [5 marks] Establish a bound implying vanishing state sensitivity as m grows when $|w| < 1$. Then consider

$$w = 2, \quad u = 1, \quad h_0 = \tanh(2), \quad x_t = 2 - 2\tanh(2) \quad \text{for every } t.$$

Verify the states and compute $\partial h_m/\partial h_0$, holding these input values fixed. Explain how this refutes "$|w| > 1$ guarantees exploding gradients." Contrast it with

$$h_0 = 0, \qquad x_t = 0 \text{ for every } t, \qquad w = 2, \qquad u = 1.$$

Calculate both the state sensitivity and the two parameter gradients in this second case. What does the comparison show about state sensitivity versus parameter gradients?

(c) [3 marks] Consider the displayed state updates of a long short-term memory (**LSTM**) cell and a gated recurrent unit (**GRU**):

$$\text{LSTM:} \quad c_t = f_t c_{t-1} + i_t \widetilde{c}_t,$$
$$\text{GRU:} \quad h_t = (1 - z_t) h_{t-1} + z_t \widetilde{h}_t.$$

For the LSTM, c_t is the cell state, f_t the forget gate, i_t the input gate, and $\widetilde{c}_t$ the candidate cell value. For this GRU convention, z_t is the update gate and $\widetilde{h}_t$ the candidate hidden state. Gates take values between 0 and 1. Treat these equations as scalar, or consider one coordinate of a vector state.

For the **direct state path**, follow only the displayed connection from the previous state, treating gates and candidates as constants. Give its derivative factors multiplied across time, and specify which gate settings preserve this path. Explain why these products are not generally the total derivatives of a complete gated network, and why gating mitigates rather than guarantees removal of vanishing gradients.

Deliverables for Question 6

Crowdmark Q6: Parts (a)–(c): shared-parameter gradients, sensitivity and bound; calculations for both supplied trajectories; and the gate-path products and their limitations.

Code / Learn: No code or separate Learn file is required for Question 6.

Question 7: A small controlled optimization experiment

[22 marks]

Use the supplied `data.py` to generate 256 training, 128 validation, and 512 test observations in three dimensions. Fit each feature's mean and population standard deviation on training inputs only, using variance divisor 256. Subtract those means and divide by those standard deviations on every split.

Train a **multilayer perceptron (MLP)** with three inputs, 16 tanh hidden units, and one output score. The output **logit** z is converted to a class-1 probability by

$$p = \frac{1}{1 + e^{-z}}.$$

This task uses labels $y \in \{0, 1\}$. Train with mean binary cross-entropy:

$$\ell = -\frac{1}{b} \sum_{i=1}^{b} \left[y_i \log p_i + (1 - y_i) \log(1 - p_i) \right]$$

on a mini-batch of b examples. Use the supplied NumPy engine (`model.py`), which evaluates this loss stably from logits and provides forward/backward calculations, optimizer updates, and model-state saving/restoring. Write the experiment loop, logging, selection, plotting, and diagnostics. You need not implement BatchNorm derivatives.

Fixed protocol. A *configuration* is one choice of BatchNorm on/off and optimizer (plain SGD or momentum SGD). A *run* trains one configuration with one random seed. Use

$$\eta = .1, \qquad \text{seeds } 11, 22, \qquad 60 \text{ epochs}, \qquad b = 32, \qquad \text{weight decay } = 0.$$

This gives **four configurations and exactly eight runs**. One epoch is one pass through the training set. An update uses one mini-batch, so each full epoch has eight updates and each complete run has 480 updates. Momentum follows

$$\mathbf{v}_{t+1} = \beta \mathbf{v}_t + \mathbf{g}_t, \qquad \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \mathbf{v}_{t+1}, \qquad \mathbf{v}_0 = \mathbf{0},$$

where $\boldsymbol{\theta}$ collects trainable parameters and $\mathbf{g}_t$ is the current mini-batch gradient. Set $\beta = .9$ for momentum and $\beta = 0$ for plain SGD.

Initialize weights independently as zero-mean normals with variance 1/fan-in, where fan-in is the number of inputs to a unit; initialize biases at zero. For each seed $s \in \{11, 22\}$, call the supplied `affine_initial_state(s)` once. Separately create `shuffle_rng = np.random.default_rng(s)` once and generate the 60 epoch orders by 60 successive calls to `shuffle_rng.permutation(256)`. Reuse copies of the initial affine state and the same saved orders across all four configurations. Divide each order into eight consecutive mini-batches of 32. Do not reset the shuffle generator between epochs. The initialization function uses its own generator, so drawing weights does not advance the shuffle generator. No dropout or augmentation is used.

BatchNorm settings. When enabled, BatchNorm is between the hidden weighted sum plus bias and tanh. Initialize its learned scale to one, learned shift to zero, running mean to zero, and running variance to one. Use $\varepsilon = 10^{-5}$ and current-batch variance with divisor b. Update each running statistic by

$$\text{new running statistic} = 0.9\,(\text{old running statistic}) + 0.1\,(\text{current batch statistic}).$$

Use divisor b also for the variance in this running update. **Evaluation mode** uses stored running statistics without changing them; **training mode** uses current-batch statistics and updates the running statistics.

Logging and checkpoints. After every epoch, record training and validation loss and accuracy in evaluation mode. A *checkpoint* is a saved copy of the trainable parameters and BatchNorm running statistics. Retain the checkpoint with smallest validation loss; an exact tie favors the earlier epoch. "Best checkpoint" below always means this loss-based choice, not the epoch with highest accuracy.

Stop and flag a run if a loss or model state becomes nonfinite (NaN or infinity); record its failure epoch and retain its best earlier finite checkpoint, if any. A configuration is eligible only if both seeds have finite checkpoint losses. Do not retune or replace failed runs. Follow the missing-output and diagnostic fallback rules in Appendix A (page 12).

Keep the explanatory discussion for parts (a)–(d) to **at most 600 words in total**, excluding tables, the figure, and brief captions. This is a maximum, not a target; do not repeat all supplied protocol settings.

(a) [5 marks] Before inspecting results, state **one hypothesis** about whether BatchNorm changes momentum's effect relative to plain SGD. Specify a numerical *paired comparison* (comparing methods within the same seed, initialization, and training order) whose outcome could support or contradict it. Identify the other variables held fixed and implement the reproducible protocol.

In one sentence, explain why equal nominal learning rates need not produce equal update scales.

(b) [7 marks] Report every run's seed, configuration, best epoch, validation loss and accuracy at that same checkpoint, epochs completed, divergence flag, and runtime (elapsed training/evaluation time in seconds). Submit **one four-panel loss figure**, with seeds as rows and training/validation loss as columns. Use the same condition color in every panel and label the scales.

Discuss **two main observations** using numerical paired comparisons, including any disagreement between seeds. There is no required method ranking.

(c) [5 marks] Select one configuration by averaging its two best-checkpoint validation losses. Break exact ties by preferring no BatchNorm, then plain SGD. **Freeze the choice before inspecting test performance.** Evaluate its two selected checkpoints once on the test set. Report each test loss and accuracy and their means.

Explain the different roles of checkpoint selection, configuration selection, and final testing. State one limitation of uncertainty claims from two seeds on one data split.

(d) [5 marks] Choose the eligible BatchNorm configuration with lowest mean validation loss and use its seed-11 checkpoint; an exact tie favors plain SGD. Compare its logits on the same validation observations processed in one batch and in chunks of eight. Record

$$\max_{1 \le i \le 128} \left| z_i^{\text{whole}} - z_i^{\text{chunks}} \right|$$

first in evaluation mode and then in training mode. Here both logits refer to the same observation i; restore the original example order before comparing.

Start each of the two training-mode evaluations from a **separate fresh checkpoint copy**, without gradients or optimizer updates. Preserve the checkpoint used for testing. Explain the observed difference using the normalization statistics.
Use the fallback in Appendix A if the required checkpoint is unavailable.

Deliverables for Question 7

Crowdmark Q7: Parts (a)–(d): the prospective hypothesis and protocol; eight-row results table; one four-panel loss figure; two numerical observations; selected configuration and individual/mean test metrics; both measured BatchNorm logit differences and their interpretation. Explanatory discussion: at most 600 words total, excluding table, figure and brief captions.

Code / Learn: Combine these Q7 files with **the Q1 code and helpers** in **ONE ZIP**: `Q7.py`, `data.py`, `model.py`, any helpers, `Q7_results.csv`, `Q7_history.csv`, and `Q7_test.csv`. Include **one README covering both questions** (commands, dependencies, seeds). Name the combined file `WatIAM_A1.zip`, replacing `WatIAM` with your Waterloo username. After completing both questions, upload this single ZIP once to the **Assignment 1 Dropbox in Learn**. Do not submit separate Q1 and Q7 ZIPs or separate code files. Use the CSV columns in the appendix (page 12). Code must regenerate the figure; checkpoint and figure files need not be included. Do not paste full code into Crowdmark.

Appendix A: Q7 output formats and fallback rules

Submission README

Create one README.md or README.txt for your submission. Give the run commands, dependencies, and seeds for both Q1 and Q7. Include it with both questions' files in the single WatIAM_A1.zip submitted to Learn.

CSV conventions

Use the column names below in the listed order. Each header must be one CSV line; line breaks here are only for readability. Use none or batchnorm for normalization; sgd or momentum for optimizer; True or False for diverged. Epoch numbers start at 1; accuracies are fractions from 0 to 1. Leave unavailable values and a nonapplicable failure_epoch empty.

Q7_results.csv: eight rows, one per run

Report best_epoch and validation metrics from the same best finite checkpoint. runtime_seconds measures the training/evaluation loop, excluding figure generation.

```
seed,normalization,optimizer,learning_rate,best_epoch,validation_loss,validation_accuracy,epochs_completed,
diverged,failure_epoch,runtime_seconds
```

Q7_history.csv: one row per completed finite epoch

Record both training and validation metrics in evaluation mode. A complete experiment has 480 rows across the eight runs.

```
seed,normalization,optimizer,epoch,train_loss,train_accuracy,validation_loss,validation_accuracy
```

Q7_test.csv: two rows for the frozen selected configuration

Use one row per seed, evaluated at its selected checkpoint.

```
seed,normalization,optimizer,best_epoch,test_loss,test_accuracy
```

If a run fails

Stop if a batch/evaluation loss or model state becomes nonfinite. Model state includes parameters and BatchNorm running statistics; model.all_finite() checks both. Keep all eight results rows. Count only epochs with finite end-of-epoch metrics in epochs_completed and history; record a failing epoch in failure_epoch. Retain any earlier best finite checkpoint, even if the run later fails. If there is none, leave best_epoch and its validation metrics empty and use an infinite internal selection score. Do not retune or replace failed runs.

A configuration is eligible only if both seeds have finite checkpoint losses. If none is eligible, do not evaluate the test set: write only the Q7_test.csv header and explain the failure in your Q7 answer and your README.

Fallback for Q7(d) only

If no BatchNorm configuration is eligible for both seeds, use any available finite BatchNorm checkpoint and identify its seed, optimizer, and epoch. If none exists, state that the diagnostic is unavailable and explain analytically the expected training/evaluation-mode behavior. Do not invent measurements.