Calvin AI/ML Handout | *Feb 23, 2026*   Name: ______________________________

1. Let's use an MLP with one hidden layer to **classify** handwritten digits as 0, 1, ..., 9. We'll give the model a 28x28-pixel image of a digit, flattened into a 784-dimensional vector. Fill in the blanks below with reasonable values.

```
W1 = torch.randn(size=(______________________))
b1 = torch.randn(size=(______________________))
W2 = torch.randn(size=(______________________))
b2 = torch.randn(size=(______________________))

for x_batch, y_batch in training_data:
  # forward pass, starting with linear layer
  linear_out_1 = ______________________________
  activations_1 = _____________________________
  logits = ______________________________
  probs = ______________________________
  loss = cross_entropy_loss(probs, y_batch)

  # backward pass
  loss.backward() # grads now stored in .grad
  for param in [W1, b1, W2, b2]:
    param += _________________________
```

```
# Fill in the shapes below:
x_batch.shape = (N, 784); y_batch.shape = (N,)
linear_out_1.shape = _________________________
activations_1.shape = _______________________
logits.shape = _________________________
probs.shape = ______________________________
loss.shape = _________

W1.grad.shape = _________________________
b1.grad.shape = _________________________
W2.grad.shape = _________________________
b2.grad.shape = _________________________
```

Calvin AI/ML Handout | *Feb 23, 2026*   Name: ______________________________

1. Let's use an MLP with one hidden layer to **classify** handwritten digits as 0, 1, ..., 9. We'll give the model a 28x28-pixel image of a digit, flattened into a 784-dimensional vector. Fill in the blanks below with reasonable values.

```
W1 = torch.randn(size=(______________________))
b1 = torch.randn(size=(______________________))
W2 = torch.randn(size=(______________________))
b2 = torch.randn(size=(______________________))

for x_batch, y_batch in training_data:
  # forward pass, starting with linear layer
  linear_out_1 = ______________________________
  activations_1 = _____________________________
  logits = ______________________________
  probs = ______________________________
  loss = cross_entropy_loss(probs, y_batch)

  # backward pass
  loss.backward() # grads now stored in .grad
  for param in [W1, b1, W2, b2]:
    param += _________________________
```

```
# Fill in the shapes below:
x_batch.shape = (N, 784); y_batch.shape = (N,)
linear_out_1.shape = _________________________
activations_1.shape = _______________________
logits.shape = _________________________
probs.shape = ______________________________
loss.shape = _________

W1.grad.shape = _________________________
b1.grad.shape = _________________________
W2.grad.shape = _________________________
b2.grad.shape = _________________________
```

For next time

Open figure 2.2 of https://udlbook.github.io/udlfigures/ (linear regression with least-squares loss). Sketch the following plots by hand (accuracy isn't critical, but try to capture the shape including curvature). Use the initial values (intercept = 1.20, slope = 0.2) for each plot while you sweep the other parameter. *If you have time, repeat this with w2 from the ReLU interactive notebook*

a. Plot **intercept** on the x-axis and **loss** on the y-axis.

b. Plot **slope** on the x-axis and **loss** on the y-axis.

2. On both plots above, mark the points corresponding to intercept = 1.20 and slope = 0.2. Sketch the tangent lines at those points and use them to compute estimates of $\frac{\delta \text{ loss}}{\delta \text{ intercept}}$ and $\frac{\delta \text{ loss}}{\delta \text{ slope}}$. Write down your estimates below. Be prepared to discuss next class what this tells you about how you might adjust the parameters to reduce the loss.

For next time

Open figure 2.2 of https://udlbook.github.io/udlfigures/ (linear regression with least-squares loss). Sketch the following plots by hand (accuracy isn't critical, but try to capture the shape including curvature). Use the initial values (intercept = 1.20, slope = 0.2) for each plot while you sweep the other parameter. *If you have time, repeat this with w2 from the ReLU interactive notebook*

a. Plot **intercept** on the x-axis and **loss** on the y-axis.

b. Plot **slope** on the x-axis and **loss** on the y-axis.

2. On both plots above, mark the points corresponding to intercept = 1.20 and slope = 0.2. Sketch the tangent lines at those points and use them to compute estimates of $\frac{\delta \text{ loss}}{\delta \text{ intercept}}$ and $\frac{\delta \text{ loss}}{\delta \text{ slope}}$. Write down your estimates below. Be prepared to discuss next class what this tells you about how you might adjust the parameters to reduce the loss.