Data 188 Introduction to Deep Learning
Spring 2026 Eric Kim

Midterm Exam (Spring 2026)

PRINT your student ID: _____

PRINT AND SIGN your name: _____, _____ _____
(last) (first) (signature)

Section 0: Pre-exam questions

1. Honor Code: Please copy the following statement in the space provided below and sign your name below.

As a member of the UC Berkeley community, I act with honesty, integrity, and respect for others. I will follow the rules and do this exam on my own.

2. What's your favorite thing about this semester?

 All Rights Reserved. This may not be publicly shared without explicit permission.

PRINT your initials and student ID: _____

3. Bag of Questions (15 pts)

(a) (3 pts) Suppose we used Sigmoid as an activation function instead of relu. What could go wrong with this? Hint: consider what happens in backprop when the input activations to sigmoid have large magnitude (or large negative magnitude).

- ◯ **A**: The model can suffer from vanishing gradients, leading to model instability
- ◯ **B**: The model can suffer from vanishing gradients, leading to slow model training
- ◯ **C**: The model can suffer from exploding gradients, leading to model instability
- ◯ **D**: The model can suffer from exploding gradients, leading to slow model training

Solution: B. Sigmoid can suffer from vanishing gradient problem when inputs have large magnitude.

(b) (3 pts) Suppose instead of CrossEntropyLoss, I want to train an MNIST digit deep learning classifier using the 0/1 classification loss, defined as:

$$L_{0/1}(\hat{y}, y) = I(\hat{y} \neq y)$$

Where $\hat{y}$ is the predicted class label, y is the ground-truth label, and $L_{0/1}(\hat{y}, y)$ is 1 if $(\hat{y} \neq y)$, 0 otherwise.

Will this work? In other words, will a model successfully train and perform at least adequately?

- ◯ **A**: Yes it will work, but CrossEntropyLoss is likely a better choice because it's designed for classification.
- ◯ **B**: No it will not work, because 0/1 loss is not differentiable.
- ◯ **C**: No it will not work, because 0/1 loss can only handle 1 class (binary classification).
- ◯ **D**: No it will not work, because 0/1 loss is not designed to operate on image inputs.

Solution: B

(c) (3 pts) Suppose I wanted to use Mean Squared Error (MSE) as the loss function to train an MNIST digit deep learning classifier:

$$L_{\text{MSE}}(\hat{y}, y) = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

Will this work? In other words, will a model successfully train and perform at least adequately?

- ◯ **A**: Yes it will work, but CrossEntropyLoss is likely a better choice because it's designed for classification.
- ◯ **B**: No it will not work, because MSE loss is not differentiable.
- ◯ **C**: No it will not work, because MSE loss can only handle 1 class (binary classification).
- ◯ **D**: No it will not work, because MSE loss is not designed to operate on image inputs.

Solution: A.

 All Rights Reserved. This may not be publicly shared without explicit permission.

PRINT your initials and student ID: _____

(d) (3 pts) Any linear operator f mapping $\mathbb{R}^n \rightarrow \mathbb{R}^m$ can be represented as a matrix A with $shape = [m, n]$, expressed as $f(x) = Ax$

◯ **A**: Always True
◯ **B**: Sometimes True (counter examples exist, it depends on the linear operator)
◯ **C**: Never True

Solution: Answer: A.

(e) (3 pts) Suppose we have a Multi-Layer Perceptron classifier, consisting of alternating $Linear$ and $Relu$ layers. Suppose we modified Relu to be $max(-k \cdot x, x)$ for some hyper parameter k (see Figure 1). As we vary k from 0.0 to 1.0, what impact to model capacity (aka model complexity) do you expect to see?

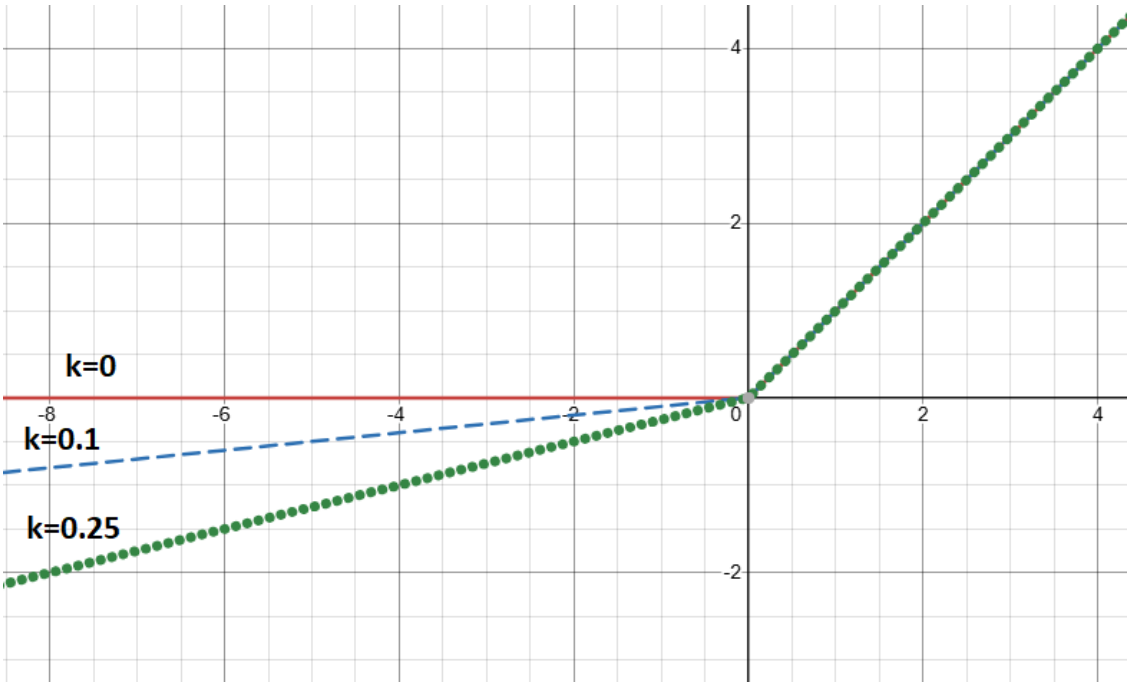

Figure 1: A visualization of $max(-k \cdot x, x)$ as we vary k from 0.0 to 1.0 (up to 0.25 is shown). Note that when $k = 0$, this is equivalent to the original $Relu$ function.

◯ **A**: As k approaches 1, model capacity will decrease.
◯ **B**: As k approaches 1, model capacity will increase.
◯ **C**: The value of k has no impact on model capacity.

Solution: A. When k becomes 1, Relu becomes a no-op (identity function), which means your NN has no nonlinearities, reducing your model capacity to just a single linear layer.

4. My Little Perceptron (MLP) (20 pts)

(a) (10 pts) Consider this Multi-Layer Perceptron (MLP) classification model (Figure 2):
Let input x have shape=[batchsize, 2], ground truth classification labels y have shape=[batchsize]. There are 20 classes. Assume all Linear have a bias, and all LayerNorm layers use affine scale and shift.

 All Rights Reserved. This may not be publicly shared without explicit permission.

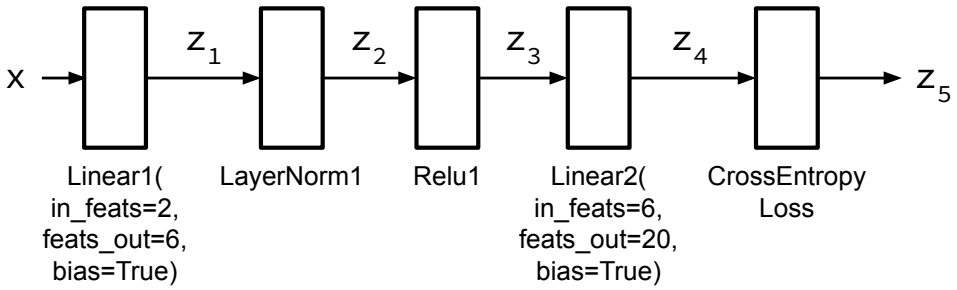

Figure 2: A Multi-Layer Perceptron (MLP) classification model.

Count the number of trainable model parameters in this model, layer by layer. If a layer has no trainable model parameters, write "0".

Linear1: _____ Linear2: _____

LayerNorm1: _____ CrossEntropyLoss: _____

Relu1: _____

Solution: Linear1: 2*6 + 6 = 18
LayerNorm1: 6 + 6 = 12
Dropout1: 0
Relu1: 0
Linear2: 6*20 + 20 = 140
CrossEntropyLoss: 0

(b) (10 pts) Write the shapes of all intermediate activations z_i. Include the "batchsize" in your answers (you may use B as shorthand). Assume that CrossEntropyLoss calculates the average loss across the entire batch, outputting a scalar.

x: shape=[B, 2] z_3: _____

z_1: _____ z_4: _____

z_2: _____ z_5: _____

Solution: z_1: [batchsize, 6]
z_2: [batchsize, 6]
z_3: [batchsize, 6]

 All Rights Reserved. This may not be publicly shared without explicit permission.

z_4: [batchsize, 20]
z_5: [] ([1] is also accepted).

5. Backpropagation (30 pts)

(a) (2 pts) Let $y = f(x_1, x_2, x_3) = (x_1 + x_2)^2 + x_2x_3 - x_3$.
You will fill in the boxes to denote the correct mathematical operation in the scalar computational graph (Figure 3). x_1, x_2, x_3 are scalar values.

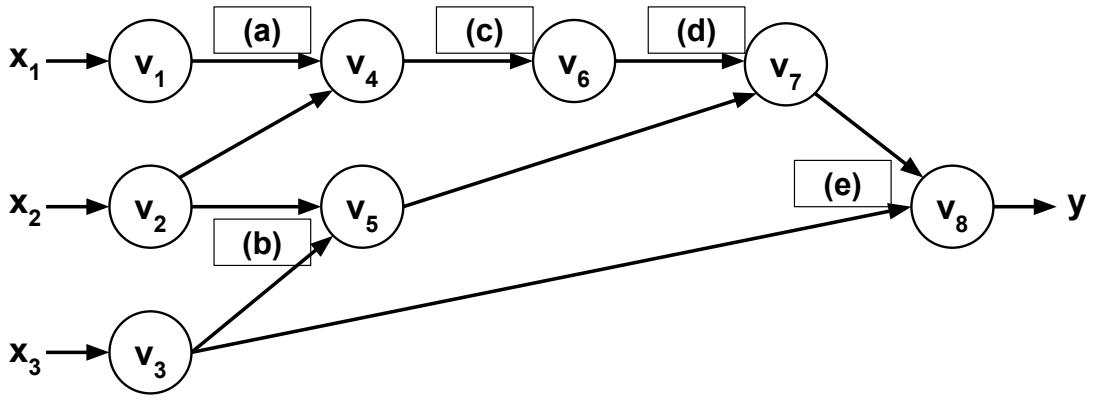

Figure 3: A computation graph.

Which operation belongs in blank (a)?

- ◯ **A**: + (summation)
- ◯ **B**: - (subtraction)
- ◯ **C**: * (multiplication)
- ◯ **D**: square (ie $square(x) = x^2$)

(b) (2 pts) Which operation belongs in blank (b)?

- ◯ **A**: + (summation)
- ◯ **B**: - (subtraction)
- ◯ **C**: * (multiplication)
- ◯ **D**: square (ie $square(x) = x^2$)

(c) (1 pts) Which operation belongs in blank (c)?

- ◯ **A**: + (summation)
- ◯ **B**: - (subtraction)
- ◯ **C**: * (multiplication)
- ◯ **D**: square (ie $square(x) = x^2$)

(d) (1 pts) Which operation belongs in blank (d)?

- ◯ **A**: + (summation)
- ◯ **B**: - (subtraction)
- ◯ **C**: * (multiplication)
- ◯ **D**: square (ie $square(x) = x^2$)

(e) (2 pts) Which operation belongs in blank (e)?

- ◯ **A**: + (summation)
- ◯ **B**: - (subtraction)
- ◯ **C**: * (multiplication)
- ◯ **D**: square (ie $square(x) = x^2$)

Solution: (a) A. (+)
(b) C (*)
(c) D (square)
(d) A (+)
(e) B (-)

 All Rights Reserved. This may not be publicly shared without explicit permission.

PRINT your initials and student ID: _____

(f) (10 pts) Consider this computation graph, where input x has shape=[d], ground truth y has shape=[d], and MSE denotes Mean Square Error. We will perform backpropagation on this graph (Figure 4).

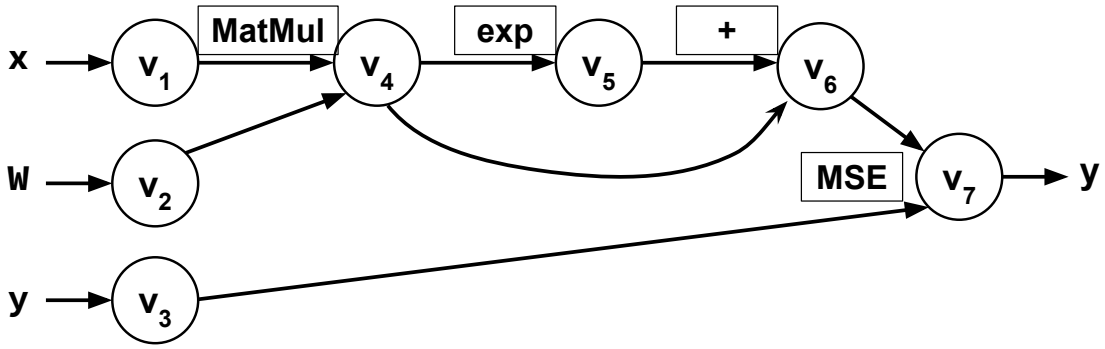

Figure 4: Another computation graph.

We've expressed the above computation graph as:

$$v_1 = x \qquad v_5 = exp(v_4)$$
$$v_2 = W \qquad v_6 = v_4 + v_5$$
$$v_3 = y \qquad v_7 = MSE(v_6, v_3)$$
$$v_4 = v_1 @ v_2$$

First, fill in the below blanks to express each adjoint $\overline{v_i}$ in terms of (1) other adjoints $\overline{v_j}$, and (2) partial derivative terms $\frac{\partial v_i}{\partial v_j}$. **Don't calculate the terms yet, leave your answers symbolically as described in the previous sentence.** The first few have been done for you. Use "@" for matrix multiplication.

$\overline{v_7} = 1$

$\overline{v_6} = \overline{v_7} @ \frac{\partial v_7}{\partial v_6}$

$\overline{v_5} =$ _____

$\overline{v_4} =$ _____

$\overline{v_3} =$ _____

$\overline{v_2} =$ _____

$\overline{v_1} =$ _____

 © UCB Data 188, Spring 2026. All Rights Reserved. This may not be publicly shared without explicit permission.

Solution: $\overline{v_7} = 1$

$$\overline{v_6} = \overline{v_7} @ \frac{\partial v_7}{\partial v_6}$$

$$\overline{v_5} = \overline{v_6} @ \frac{\partial v_6}{\partial v_5}$$

$$\overline{v_4} = \overline{v_5} @ \frac{\partial v_5}{\partial v_4} + \overline{v}_6 @ \frac{\partial v_6}{\partial v_4}$$

$$\overline{v_3} = \overline{v_7} @ \frac{\partial v_7}{\partial v_3}$$

$$\overline{v_2} = \overline{v_4} @ \frac{\partial v_4}{\partial v_2}$$

$$\overline{v_1} = \overline{v_4} @ \frac{\partial v_4}{\partial v_1}$$

 © UCB Data 188, Spring 2026. All Rights Reserved. This may not be publicly shared without explicit permission.

PRINT your initials and student ID: _____

(For your convenience, the v_i is copied below from the previous page)

$$v_1 = x \qquad v_5 = exp(v_4)$$
$$v_2 = W \qquad v_6 = v_4 + v_5$$
$$v_3 = y \qquad v_7 = MSE(v_6, v_3)$$
$$v_4 = v_1 @ v_2$$

(g) (12 pts) Next, calculate the adjoints $\overline{v_i}$. **Calculate each term to be only in terms of v_i.** Your final answer should not have any adjoints $\overline{v_i}$ or partial derivative $\frac{\partial v_i}{\partial v_j}$ terms remaining. Please do not evaluate terms to include x, W, or y: your answers should only consist of operations involving v_i (and possibly constants). **Please be explicit about distinguishing between matrix multiplication ("@") and elementwise multiplication ("*").**
The following facts may be useful:

For $z = A@B$, we have:

$$\frac{\partial loss}{\partial z} @ \frac{\partial z}{\partial A} = \frac{\partial loss}{\partial z} @ B^T$$
$$\frac{\partial loss}{\partial z} @ \frac{\partial z}{\partial B} = A^T @ \frac{\partial loss}{\partial z}$$

For $z = \text{MSE}(x, y) = \frac{1}{d}\sum_{i=1}^{d}(x[i] - y[i])^2$:

$$\frac{\partial z}{\partial x} = \frac{2}{d}(x - y)$$
$$\frac{\partial z}{\partial y} = -\frac{2}{d}(x - y)$$

$\overline{v_7} = 1$

$\overline{v_6} =$ _____

$\overline{v_5} =$ _____

$\overline{v_4} =$ _____

$\overline{v_3} =$ _____

$\overline{v_2} =$ _____

$\overline{v_1} =$ _____

 All Rights Reserved. This may not be publicly shared without explicit permission.

Solution: $\overline{v_7} = 1$

$\overline{v_6} = \frac{2}{d}(v_6 - v_3)$

$\overline{v_5} = \frac{2}{d}(v_6 - v_3)$

$\overline{v_4} = \frac{2}{d}(v_6 - v_3)@exp(v_4) + \frac{2}{d}(v_6 - v_3)$

$\overline{v_3} = -\frac{2}{d}(v_6 - v_3)$

$\overline{v_2} = v_1^T@(\frac{2}{d}(v_6 - v_3)@exp(v_4) + \frac{2}{d}(v_6 - v_3))$

$\overline{v_1} = (\frac{2}{d}(v_6 - v_3)@exp(v_4) + \frac{2}{d}(v_6 - v_3))@v_2^T$

6. Optimization (5 pts)

(a) (5 pts) Recall that the Adam optimizer uses these update equations:

$$m_{t+1} = \beta_1 m_t + (1 - \beta_1) g_t, \qquad v_{t+1} = \beta_2 v_t + (1 - \beta_2) g_t^2, \qquad \theta_{t+1} = \theta_t - \alpha \frac{m_{t+1}}{\sqrt{v_{t+1}} + \epsilon},$$

where $g_t = \frac{d}{d\theta} J(\theta_t)$.
Suppose we modify Adam by removing the second-moment term entirely, ie we set $v_{t+1} = 0$ for all t, and update parameters using this update rule:

$$m_{t+1} = \beta_1 m_t + (1 - \beta_1) g_t, \qquad \theta_{t+1} = \theta_t - \alpha m_{t+1}.$$

Which optimizer does this most resemble?

- ◯ **A**: "vanilla" Stochastic Gradient Descent (SGD)
- ◯ **B**: SGD + Momentum
- ◯ **C**: SGD + Momentum (with bias correction)
- ◯ **D**: SGD + Nesterov acceleration
- ◯ **E**: Newton's Method

Solution: B.

7. Parameter Initialization (12 pts)

Select **exactly one choice** for the following multiple choice questions.

 All Rights Reserved. This may not be publicly shared without explicit permission.

(a) (3 pts) For a Linear layer (with Relu), recall that Kaiming normal initialization initializes the weight parameters by randomly sampling from $\mathcal{N}(0, 2/n)$, where n is the number of input features.

What is the primary motivation for carefully choosing the variance to be $2/n$? Choose one option.

- ◯ **A**: To have the output activation magnitude match the input activation magnitude.
- ◯ **B**: To maintain enough diversity in the random sample.
- ◯ **C**: To reduce computation
- ◯ **D**: To avoid numerical issues (eg overflow or underflow).
- ◯ **E**: To reduce internal covariate shift.

Solution: A.

By having the output activation magnitude match the input activation magnitude, we're stabilizing the activation norm at each layer, which helps avoid exploding/vanishing gradients.

(b) (3 pts) Recall that, for an MLP (consisting of repeated Linear -> Relu blocks), we saw that different parameter initialization strategies for the Linear weight parameters could lead to vanishing and/or exploding gradients.

Interestingly, there are other ways to mitigate this vanishing/exploding gradients problem.

Which of these techniques will **NOT** effectively mitigate **BOTH** the vanishing and exploding gradient problem? Choose only one.

- ◯ **A**: Adam
- ◯ **B**: Activation Normalization
- ◯ **C**: Reducing the learning rate
- ◯ **D**: Residual ("skip") connections

Solution: C.

Adam partially mitigates vanishing gradients (particularly for sparse gradients) by scaling the gradient via its inverse gradient magnitude history: parameters with small gradient magnitude (and, particularly, having a history of small gradient magnitude) will have their gradients upweighted. Adam also partially mitigates exploding gradients by scaling gradients by the inverse of its gradient magnitude: parameters with large gradient magnitude have their gradients downweighted.

Activation normalization mitigates the vanishing/exploding gradient problem by ensuring that intermediate activations are all at the same magnitude scale. Recall that most adjoint calculations involve a multiplication between "out_grad" and the node's inputs. Thus, it's in our best interests to ensure that the magnitude of the node's inputs are as "well-behaved" as possible.

Reducing the learning rate can partially mitigate the exploding gradient problem, but it won't help the vanishing gradient problem (in fact, it'll make it worse!).

Residual ("skip") connections mitigate the vanishing gradient problem by allowing "distant" adjoints to flow directly to the current adjoint, without having to go through many repeated multiplications. Skip connections do partially mitigate the exploding gradient problem, but the argument is less direct. One may wonder why skip connections mitigate the exploding gradient problem at all, as the adjoints of the lower layers of deep networks still contain multiplications of many terms, eg from the "non-skip" path.

One argument is that skip connections better-condition the learning problem to learn a function that is easier and less-prone to exploding gradients. This is (somewhat indirectly) supported by the observation that adding skip connections makes the loss landscape much smoother and easier to optimize over: https://arxiv.org/pdf/1712.09913. If you're curious, this paper goes into an interesting

 All Rights Reserved. This may not be publicly shared without explicit permission.

deep-dive into exploding gradients and its connection to skip connections, though out-of-scope for this course: https://arxiv.org/abs/1712.05577

 © UCB Data 188, Spring 2026. All Rights Reserved. This may not be publicly shared without explicit permission.

PRINT your initials and student ID: _____

(c) (3 pts) Consider the following classification model architecture (Figure 5):

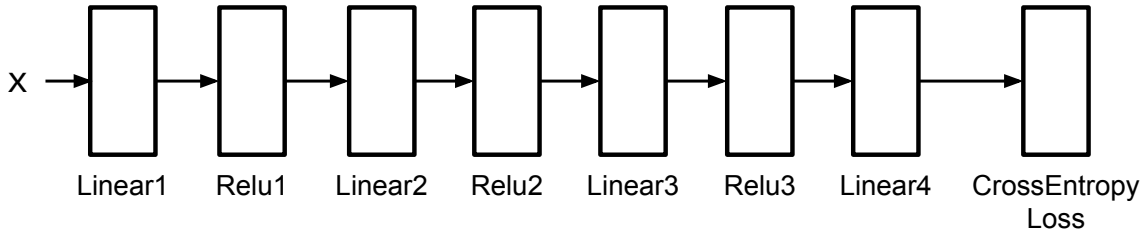

Figure 5: Yet another MLP model architecture.

Suppose I initialized the weight parameters of all Linear layers to all 0's. Assume the Linear layers have no bias. What will happen? Choose only one choice.

- ◯ **A**: The model will train fine.
- ◯ **B**: The model will train slowly
- ◯ **C**: The model will have unstable training, and may diverge
- ◯ **D**: The model will fail to learn anything

Solution: D.

The gradient updates for the weight parameters will all be 0.

(d) (3 pts) In Figure 5, suppose I add a bias term to all Linear layers. Suppose I initialize the weight parameters of all Linear layers via an appropriate initialization strategy, and I initialize the bias parameters of all Linear layers to be a very large value, say $1e6$. Please explain why this may be a bad idea. Ignore issues relating to exploding gradients or numerical under/overflow.

Hint: consider the fact that we are using the Relu activation function.

- ◯ **A**: The model will suffer from the "dying Relu" problem
- ◯ **B**: The model complexity will initially collapse into a single Linear layer
- ◯ **C**: The computation cost will significantly increase
- ◯ **D**: The model will be difficult to regularize, and will be prone to overfitting.

Solution: B.

Having a high initial bias results in all Relu functions acting as a no-op linear function, which injects no nonlinearities into your model. This results in your model complexity "collapsing" into a linear model, at least at first. With enough train iterations this can eventually (hopefully!) be fixed: hopefully, the weight updates will "steer" the model into an area where Relu nonlinearities will start triggering, thus increasing the model expressiveness.

8. Normalization (15 pts)

For the following questions, please use these formulas.

For calculating mean: $mean(X) = \frac{1}{n}\sum_{i=1}^{n} x_i$.

For calculating variance: $var(X) = \frac{1}{n}\sum_{i=1}^{n}(x_i - mean(x))^2$.

 All Rights Reserved. This may not be publicly shared without explicit permission.

(a) (6 pts) Perform LayerNorm1d on this input X with $shape = [batchsize = 2, feats = 2]$. Use $eps = 0.0$. Ignore the affine shift parameters. You may leave your answer as simplified fractions/radicals (if any). Show your work.

$$X = \begin{bmatrix} 8 & 2 \\ 0 & 4 \end{bmatrix}$$

Solution:

$$\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

(b) (6 pts) Perform BatchNorm1d on this input X with shape=$[batchsize = 2, feats = 3]$. Use $eps = 0.0$. Ignore the affine shift parameters. You may leave your answer as simplified fractions/radicals (if any). Show your work.

$$X = \begin{bmatrix} 2 & 3 & 4 \\ 1 & 2 & 2 \end{bmatrix}$$

Solution:

$$\begin{bmatrix} 1 & 1 & 1 \\ -1 & -1 & -1 \end{bmatrix}$$

 © UCB Data 188, Spring 2026. All Rights Reserved. This may not be publicly shared without explicit permission.

PRINT your initials and student ID: _____

(c) (3 pts) Suppose I trained a model with BatchNorm, but I forgot to enable training=True for the BatchNorm layers. Assume BatchNorm does not have the learnable affine parameters (scale, shift). Assume running_mean is initialized to 0, running_var is initialized to 1, and eps=0.0.

Describe how BatchNorm will behave during training.

◯ **A**: BatchNorm will correctly normalize input features to have mean=0, variance=1, but will fail to update running_mean, running_var.

◯ **B**: BatchNorm will throw an error (ex: division by zero, under/overflow, etc)

◯ **C**: BatchNorm will behave as an identity function, passing the input through unchanged (ie a "no-op")

◯ **D**: BatchNorm will return a constant value (ex: all 0's, all 1's, etc).

Solution: C. In eval mode, BN behaves as $(x - running\_mean)/sqrt(running\_var)$. When running_mean, running_var are left at its initial values (0, 1), this reduces to: $(x - 0)/sqrt(1) = x$.

9. Regularization (10 pts)

```
def dropout(x: Tensor, p: float, training: bool) -> Tensor:
    # x.shape=[batchsize, d]
    # Drop probability = p
    # mask is a matrix of same shape as x, with values:
    #   mask[i][j] = 0 -> drop value at x[i][j]
    #   mask[i][j] = 1 -> keep value at x[i][j]
[1]    mask = Bernoulli(1 - p).sample(
[2]        shape = x.shape
[3]    )
[4]    if training:
[5]        return (mask * x) / (1 - p)
[6]    else:
[7]        return (mask * x)
```

(a) (3 pts) I claim there is an issue in the implementation. Please identify the bug.

Hint: assume the 'Bernoulli(1 - p)' sampling is correct.

◯ **A**: In Line 2, the shape of 'mask' is incorrect.

◯ **B**: In Line 5, we should divide by 'p', not '(1 - p)'

◯ **C**: In Line 7, we should divide by 'p'

◯ **D**: In Line 7, we should divide by '(1 - p)'

◯ **E**: In Line 7, we should return 'mask', not 'mask * x'

Solution: IMPORTANT: This question has a bug. Option E should have said: "In Line 7, we should return 'x', not 'mask * x'". In response, we granted all students full points for this question (even if they left it blank).

(with the aforementioned bugfix): E. during test time, we shouldn't mask out the input.

(b) (3 pts) For an image classification model, choose the layer that would be the poorest place to place a Dropout layer:

 All Rights Reserved. This may not be publicly shared without explicit permission.

◯ **A**: After a Conv2d layer
◯ **B**: After a BatchNorm2d layer
◯ **C**: After a Relu layer
◯ **D**: After the final Linear layer (that produces the logits)

Solution: D.

Applying dropout on predicted logits is destroying vital information for the classification task, and isn't achieving the goals of regularization.

(c) (4 pts) What is the primary reason for adding Dropout to a deep learning model (as discussed in this class)?

◯ **A**: To reduce computation time
◯ **B**: To avoid numerical underflow/overlow
◯ **C**: To improve generalization of the model on unseen data
◯ **D**: To reduce internal covariate shift

Solution: C.

Dropout is primarily a regularization method. Regularization is a method to improve generalization on unseen data, by avoiding overfitting on the training dataset.

10. Convnets (14 pts)

(a) (2 pts) Suppose we have an input image with dimensions 32 x 32. If we perform a 2D convolution with padding = 1, stride = 2, and kernel size = 3, what will the output dimensions be? Show your work.
Hint: for input spatial size z, padding p, stride s, kernel size k, the output spatial size is:

$$floor(\frac{z+2p-k}{s})+1$$

$height_{out} =$ _____

$width_{out} =$ _____

Solution: Using the formula $floor(\frac{W+2p-K}{stride})+1$: $height_{out} = 16$
$width_{out} = 16$

 All Rights Reserved. This may not be publicly shared without explicit permission.

PRINT your initials and student ID: _____

(b) (8 pts) Given the input image and kernel, perform a 2D convolution with: Padding = 1, Stride = 1. Use zero-padding.

Write your final answer in the boxes provided below. Start at the top left corner. Not all boxes may be used; leave unused spaces blank. Show your work.

Input image:

2	4	5
2	1	3
3	1	5

Kernel:

1	0
2	-1

Output (Fill me in!)

Solution:
$$\begin{bmatrix} -2 & 0 & 3 & 10 \\ -2 & 5 & 3 & 11 \\ -3 & 7 & -2 & 13 \\ 0 & 3 & 1 & 5 \end{bmatrix}$$

 All Rights Reserved. This may not be publicly shared without explicit permission.

PRINT your initials and student ID: _____

(c) (4 pts) Given the input image, perform max pooling with: No padding, Stride = 2, Kernel size = 2. Write your final answer in the boxes provided below. Start at the top left corner. Not all boxes may be used; leave unused spaces blank. Show your work.

Input image:

2	4	5	0
2	1	3	1
3	1	5	0
0	1	0	1

Output (Fill me in!)

Solution: $\begin{bmatrix} 4 & 5 \\ 3 & 5 \end{bmatrix}$

11. Computer Vision (25 pts)

(a) (6 pts) Consider the following convnet illustrated in Figure 6:

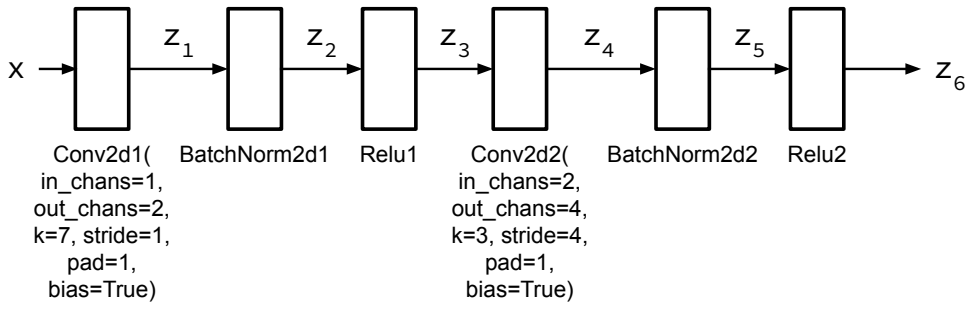

Figure 6: A convolutional neural network ("convnet").

Let input x contain grayscale (single channel) images with height=224px, width=224px, and x has shape=[batchsize, 1, 224, 224]. Assume all Conv2d layers have a bias, and all BatchNorm2d layers use affine scale and shift.

 © UCB Data 188, Spring 2026. All Rights Reserved. This may not be publicly shared without explicit permission.

Count the number of trainable model parameters in this model, layer by layer. If a layer has no trainable model parameters, write "0".

Conv2d1: _____ Conv2d2: _____

BatchNorm2d1: _____ BatchNorm2d2: _____

Relu1: _____ Relu2: _____

Solution: Conv2d1: 100
BatchNorm2d1: 4
Relu1: 0
Conv2d2: 76
BatchNorm2d2: 8
Relu2: 0

(b) (6 pts) Write the shapes of all intermediate activations z_i (include "batchsize", you may use B as shorthand).

z_1: _____ z_4: _____

z_2: _____ z_5: _____

z_3: _____ z_6: _____

Solution: z_1: [batchsize, 2, 220, 220]
z_2: [batchsize, 2, 220, 220]
z_3: [batchsize, 2, 220, 220]
z_4: [batchsize, 4, 55, 55]
z_5: [batchsize, 4, 55, 55]
z_6: [batchsize, 4, 55, 55]

 © UCB Data 188, Spring 2026. All Rights Reserved. This may not be publicly shared without explicit permission.

PRINT your initials and student ID: _____

(c) (3 pts) Suppose I wanted to perform image classification by following the technique done by the ResNet model: average each spatial feature map (transforming [B, C, H, W] to [B, C, 1, 1]) and pass this [B, C] tensor as input to the Linear layer producing the logits (with appropriate flattening) (Figure 7):

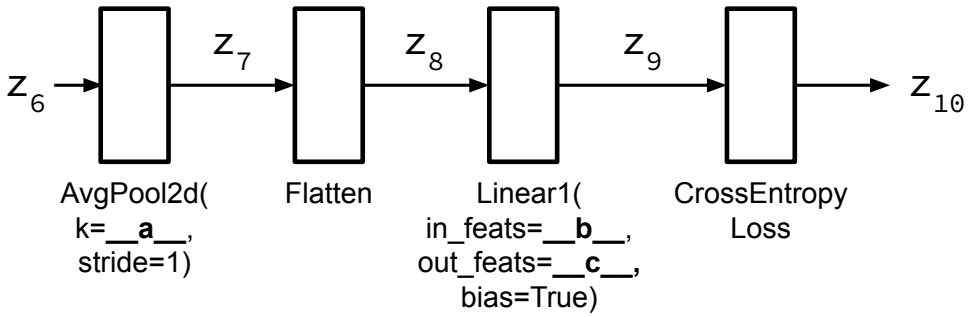

Figure 7: Adding a classifier to a convnet.

Where z_6 is the final spatial feature map from the previous Convnet architecture. For this question, assume that z_6 has shape=[batchsize, C_6, H_6, W_6], the number of classes is n_c, and write your answers in terms of these symbols (don't write your answers here in terms of numbers from the previous question). Assume that $H_6 = W_6$.

What is blank a? _____ What is blank c? _____

What is blank b? _____

Solution: a: H_6 (W_6 is also accepted) b: C_6 c: n_c

(d) (3 pts) In the YOLO object detection model, to detect N objects in an image, we need to perform N forward passes through the model.

◯ **A**: True
◯ **B**: False

Solution: False.

(e) (3 pts) At the time of its release, what were the primary strengths of the YOLO object detection model?

◯ **A**: Ability to perform simultaneous detection, classification, and segmentation.
◯ **B**: Strong ability to detect many small, tightly-grouped objects in an image
◯ **C**: Strong ability to distinguish between extremely similar objects
◯ **D**: Speed of training/inference and simplicity of approach

Solution: D.

Yolo can't do segmentation (at least, not the approach/paper we covered in class). Yolo struggles with small, tightly-grouped objects. While Yolo is able to distinguish between similar objects, this isn't a key strength of this approach.

 All Rights Reserved. This may not be publicly shared without explicit permission.

(f) (4 pts) The convnet architecture (as studied in this class) achieves invariance to what kinds of image transformations? Only focus on model architecture design, not other aspects such as data preprocessing or augmentation.

◯ **A**: Translation only
◯ **B**: Translation and rotation only
◯ **C**: Image brightness adjustments (brighten, darken, contrast) only
◯ **D**: All of the above

Solution: A.
Conv2d layers are not invariant to rotations, nor to changes in image intensity.

12. DNN Frameworks (14 pts)

(a) (5 pts) In "TensorOp.gradient(self, out_grad: Tensor, node: Tensor)", what is "out_grad"?

◯ **A**: $\frac{\partial \mathbf{loss}}{\partial \mathbf{in}}$
◯ **B**: $\frac{\partial \mathbf{loss}}{\partial \mathbf{dout}}$
◯ **C**: $\frac{\partial \mathbf{out}}{\partial \mathbf{in}}$
◯ **D**: $\frac{\partial \mathbf{in}}{\partial \mathbf{out}}$

Solution: B

(b) (4 pts) Suppose I used the following implementation for "EWiseMul.gradient()":

```
# Old (correct) implementation
class EWiseMul(TensorOp):
    """Elementwise multiplication of two ndarrays: a * b."""
    def compute(self, a: NDArray, b: NDArray) -> NDArray:
        return a * b

    def gradient(self, out_grad: Tensor, node: Tensor):
        lhs, rhs = node.inputs
        return out_grad * rhs, out_grad * lhs

# New implementation
class EWiseMul(TensorOp):
    """Elementwise multiplication of two ndarrays: a * b."""
    def compute(self, a: NDArray, b: NDArray) -> NDArray:
        return a * b

    def gradient(self, out_grad: Tensor, node: Tensor):
        lhs, rhs = node.inputs
        lhs = lhs.realize_cached_data()  # np.ndarray
        rhs = rhs.realize_cached_data()  # np.ndarray
```

 All Rights Reserved. This may not be publicly shared without explicit permission.

```
        out_grad = out_grad.realize_cached_data()  # np.ndarray
        return Tensor(out_grad * rhs), Tensor(out_grad * lhs)
```

Choose the option that best describes what happens if I use this new "EWiseMul" implementation in my needle model:

- ◯ **A**: During training, 'EWiseMul' will output incorrect gradients.
- ◯ **B**: During backpropagation, 'EWiseMul.gradient()' will infinite loop.
- ◯ **C**: Training a first-order method (like SGD) will fail with a runtime error.
- ◯ **D**: Training a first-order method (like SGD) will succeed, but second-order methods will fail.

Solution: D.

A second-order method requires running backprop on the adjoint computation graph to generate the "second-order" adjoints (as covered in this class). The above code change doesn't create the adjoint computation graph, so second-order methods won't work.

However: first-order methods (eg SGD) will be fine, since they'll have the adjoints they need to update model parameters.

 All Rights Reserved. This may not be publicly shared without explicit permission.

PRINT your initials and student ID: _____

(c) (5 pts) Recall that "MatMul" is a TensorOp that, given matrix inputs A, B, calculates: "out = A @ B" (ie matrix multiplication).

Choose the option that describes what "MatMul.gradient()" outputs:

◯ **A**: $\frac{\partial \mathbf{out}}{\partial \mathbf{A}}, \frac{\partial \mathbf{out}}{\partial \mathbf{B}}$

◯ **B**: $\frac{\partial \mathbf{out}}{\partial \mathbf{A}}, \frac{\partial \mathbf{A}}{\partial \mathbf{B}}$

◯ **C**: $\frac{\partial \mathbf{loss}}{\partial \mathbf{out}}, \frac{\partial \mathbf{out}}{\partial \mathbf{A}}$

◯ **D**: $\frac{\partial \mathbf{loss}}{\partial \mathbf{A}}, \frac{\partial \mathbf{loss}}{\partial \mathbf{B}}$

Solution: D.

 © UCB Data 188, Spring 2026. All Rights Reserved. This may not be publicly shared without explicit permission.