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?

Do not turn this page until the proctor tells you to do so. You can work on Section 0 above before time starts.

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

(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.

(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.

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

(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?

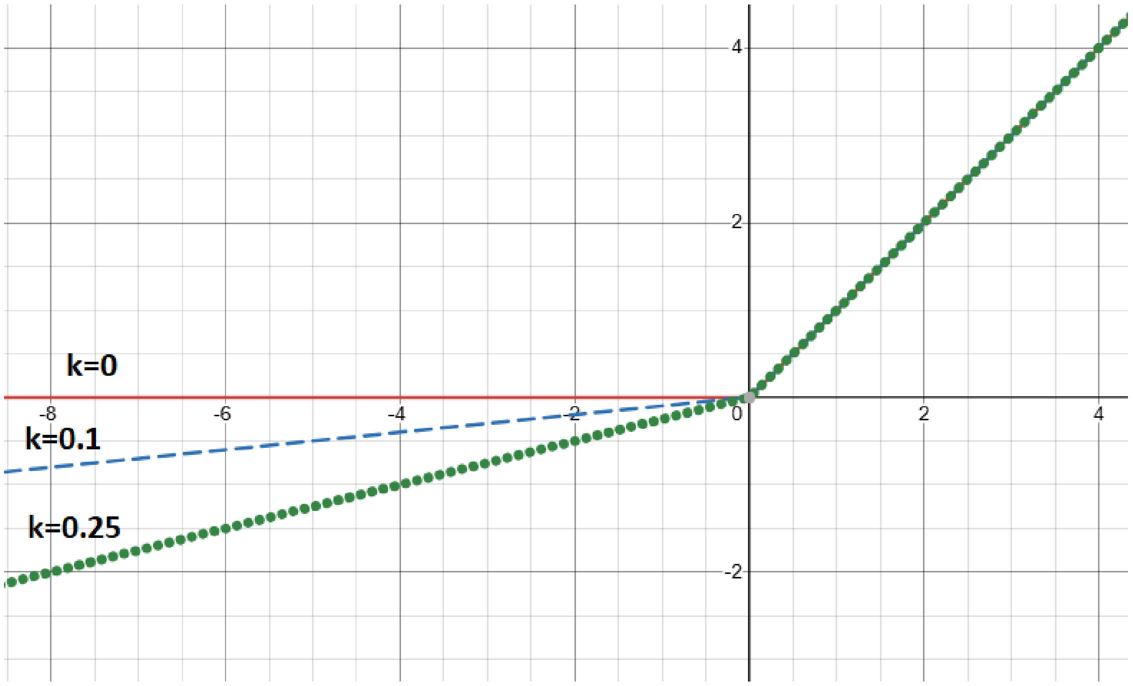

Figure 1: A visualization of $max(-k \cdot x, x)$ as we vary k from 0.0 to 1.0 (up to 0.25 is shown). Note that when $k = 0$, this is equivalent to the original $Relu$ function.

- ◯ **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.

PRINT your initials and student ID: ____________________

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

(a) (10 pts) Consider this Multi-Layer Perceptron (MLP) classification model (Figure 2):

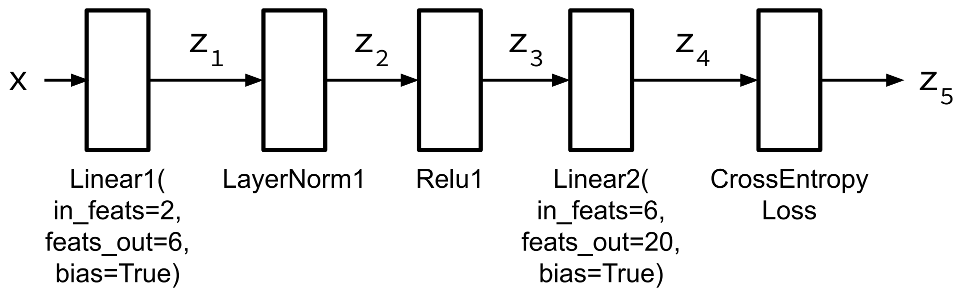
x
z1
z2
z3
z4
z5
Linear1(in_feats=2, feats_out=6, bias=True)
LayerNorm1
Relu1
Linear2(in_feats=6, feats_out=20, bias=True)
CrossEntropy Loss

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

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.

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: ____________________

(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: ____________________

PRINT your initials and student ID: ____________________

5. Backpropagation (30 pts)

(a) (2 pts) Let $y = f(x_1, x_2, x_3) = (x_1 + x_2)^2 + x_2 x_3 - x_3$.
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.

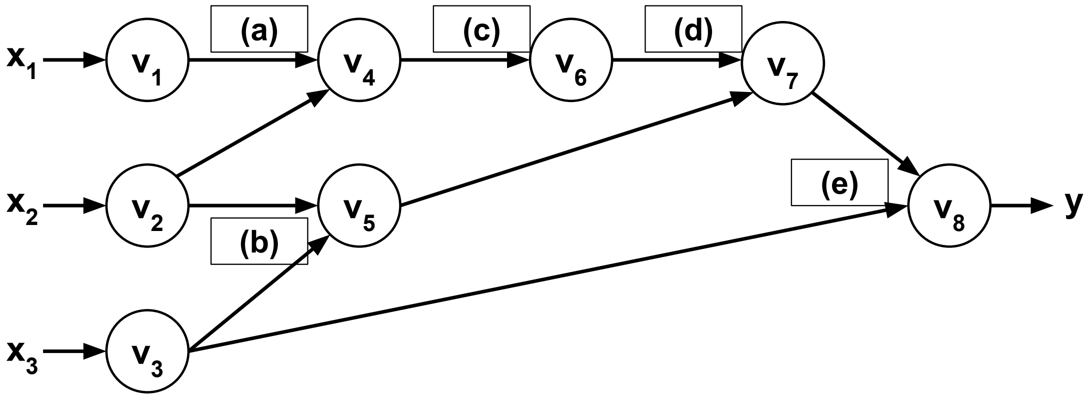

Figure 3: A computation graph.

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 $square(x) = x^2$)

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).

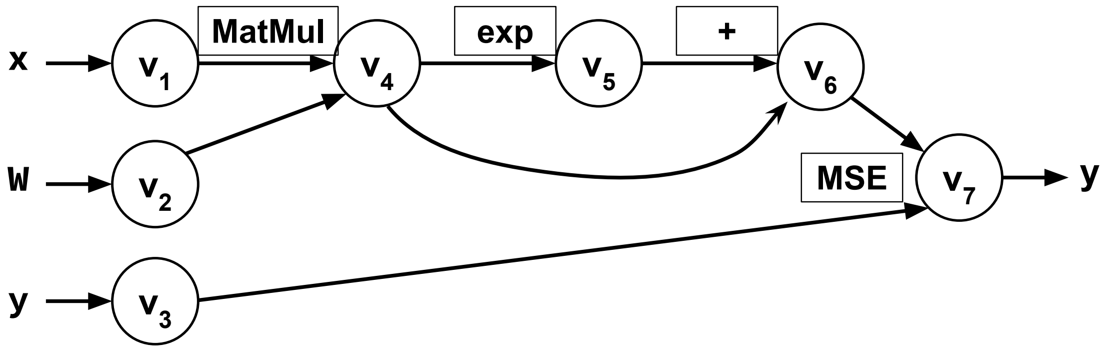

Figure 4: Another computation graph.

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.

$\overline{v_7} = 1$

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

$\overline{v_5} =$ ______________________________

$\overline{v_4} =$ ______________________________

$\overline{v_3} =$ ______________________________

$\overline{v_2} =$ ______________________________

$\overline{v_1} =$ ______________________________

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

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

$\overline{v_7} = 1$

$\overline{v_6} =$ ______________________________

$\overline{v_5} =$ ______________________________

$\overline{v_4} =$ ______________________________

$\overline{v_3} =$ ______________________________

$\overline{v_2} =$ ______________________________

$\overline{v_1} =$ ______________________________

PRINT your initials and student ID: ________________________________

[Extra page. If you want the work on this page to be graded, make sure you tell us on the problem's main page.]

PRINT your initials and student ID: ______________________________

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

PRINT your initials and student ID: ____________________

7. Parameter Initialization (12 pts)

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

(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.

(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

PRINT your initials and student ID: ____________________

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

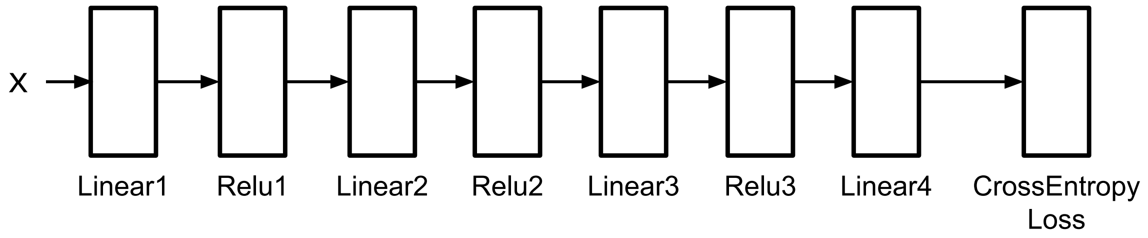

Figure 5: Yet another MLP model architecture.

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

(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.

PRINT your initials and student ID: ______________________________

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

(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.

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

(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.

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

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).

PRINT your initials and student ID: ____

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‘

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

◯ **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)

(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

PRINT your initials and student ID: ______________________________

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:

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

$height_{out} =$ ______________________________

$width_{out} =$ ______________________________

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!)

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!)

PRINT your initials and student ID: ______________________________

11. Computer Vision (25 pts)

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

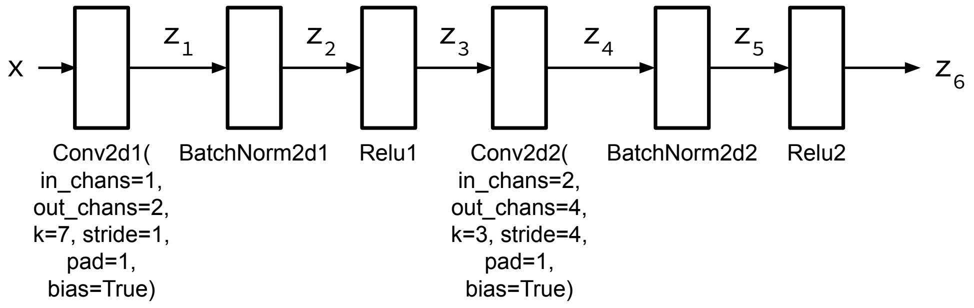
x
z_1
z_2
z_3
z_4
z_5
z_6
Conv2d1(in_chans=1, out_chans=2, k=7, stride=1, pad=1, bias=True)
BatchNorm2d1
Relu1
Conv2d2(in_chans=2, out_chans=4, k=3, stride=4, pad=1, bias=True)
BatchNorm2d2
Relu2

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.

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: ______________________

(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: ______________________

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):

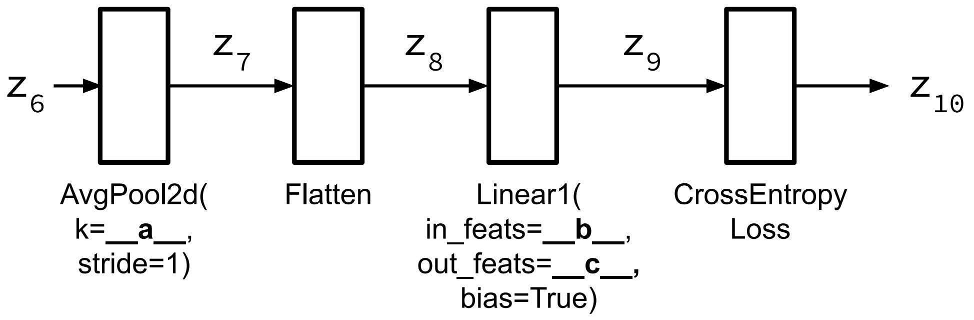
z6
z7
z8
z9
z10
AvgPool2d(k=__a__, stride=1)
Flatten
Linear1(in_feats=__b__, out_feats=__c__, bias=True)
CrossEntropy Loss

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? ____________________

(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

(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

(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

PRINT your initials and student ID: ____________________

12. DNN Frameworks (14 pts)

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

◯ **A**: $\frac{\partial \textbf{loss}}{\partial \textbf{in}}$

◯ **B**: $\frac{\partial \textbf{loss}}{\partial \textbf{dout}}$

◯ **C**: $\frac{\partial \textbf{out}}{\partial \textbf{in}}$

◯ **D**: $\frac{\partial \textbf{in}}{\partial \textbf{out}}$

(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
        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.

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

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

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

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

PRINT your initials and student ID: ____________________

[Doodle page! Draw us something if you want or give us suggestions or complaints. You can also use this page to report anything suspicious that you might have noticed.

If needed, you can also use this space to work on problems. But if you want the work on this page to be graded, make sure you tell us on the problem's main page.]