Final Exam, Fall 2024
ESE 577
SOLUTIONS

Instructor: Jorge Mendez-Mendez

Date: Thursday December 12th, 2024

Do not tear exam booklet apart!

- This is a closed book exam. One sheet (8 1/2 in. by 11 in.) of notes, front and back, are permitted. Calculators are not permitted.
- The total exam time is 2.5 hours.
- The problems are not necessarily in any order of difficulty.
- Record all your answers in the places provided. If you run out of room for an answer, continue on a blank page and mark it clearly.
- If a question seems vague or under-specified to you, make an assumption, write it down, and solve the problem given your assumption.
- If you absolutely *have* to ask a question, come to the front.
- **Write your name on every piece of paper.**

Name: ______________________ SBU email: ______________________

Question	Points
1	12
2	22
3	22
4	14
5	11
6	19
Total:	100

Name: ____________________________

Deploying a Deep Learning Model

1. (12 points) Dr. Max Pooler is a deep learning engineer at ShallowBrain and has been working on a new model to classify the rhythmic patterns of bird songs into three categories: nocturnal chirps, dawn choruses, and alarm calls, to aid in environmental monitoring. Unfortunately, Dr. Pooler cannot seem to get the model to work as well as he expected, and is calling on you to help him debug what is happening.

   (a) Dr. Pooler obtained his data from his colleague, Dr. Sigmoid, in a USB-stick containing one million files, each containing one bird song. What steps should Dr. Pooler take before starting to train his model? Choose all that apply.

   - ☐ Play each sound individually and verify that the bird sound matches the annotation in the file.
   - ☐ Hire a team of ornithology experts to analyze the data.
   - ☐ Contact multiple ornithology labs across the world to request that they share any additional data of bird songs.
   - ☐ Conduct statistical analysis on the dataset to ensure that there are no anomalies.
   - ☐ Immediately train a model to see what happens.

   ANSWER: 2, 3, 4. (1) Even if sounds last just one second, it would take a total of 278 to play all of them continuously (plus, Dr. Pooler is no expert on ornithology, so how would he even tell sounds apart?). (2) Experts might be able to listen to a representative sample and gauge whether annotations are correct. (3) ALWAYS GET MORE DATA. (4) Statistical analysis is one way to analyze the data that non-domain experts like Dr. Pooler could do. (5) We always study our data before training something.

   (b) After Dr. Pooler is certain that his data is ready to use for training, which of the following would be a reasonable common-sense baseline to beat? Choose one and briefly justify your choice.

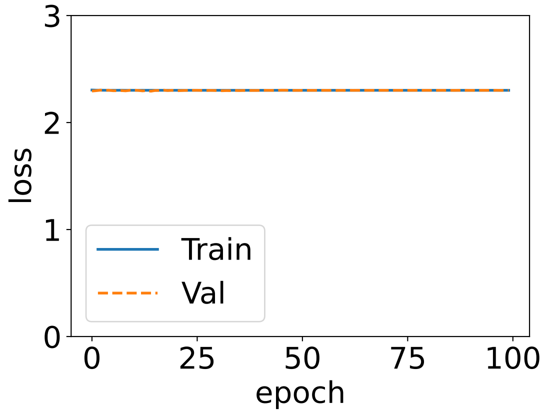
   - ○ 33%
   - ○ 90%
   - ○ 10%
   - ○ Not enough information

   Justification:

   ANSWER: Not enough information. 10% might be too little for a 3-class problem, and 90% might be too much. While 33% might seem like a good choice, without knowing whether the classes are balanced, we cannot know with certainty if 33% is a good choice.

Name: ________________________________

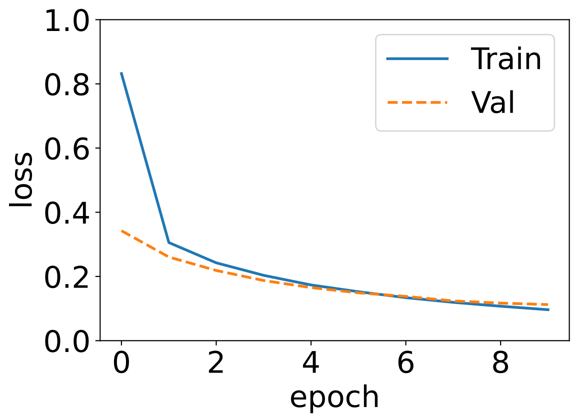
(c) After training his first model, Dr. Pooler observed the following behavior. What should Dr. Pooler do to lower his training loss?



- ☐ Train for more epochs
- ☐ Modify the learning rate
- ☐ Modify the batch size
- ☐ Increase the number of layers
- ☐ Collect more data
- ☐ Give up; this plot demonstrates that it is not possible to learn with this dataset

ANSWER: 2, 3. We can always learn something. When we can't, there is always an issue with the optimizer (learning rate or batch size are go-to hyperparameters to change).

(d) After some modifications to his model and/or data, Dr. Pooler was able to obtain the following learning curve. Which of the following steps should Dr. Pooler pursue next? Choose all that apply.



☐ Train for more epochs

☐ Modify the learning rate

☐ Modify the batch size

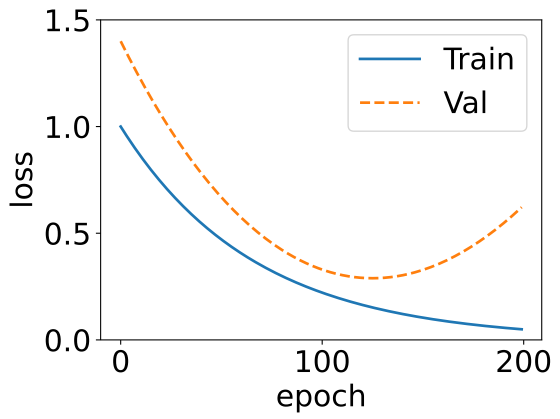
☐ Increase the number of layers

☐ Collect more data

☐ Stop here and use the model obtained at the end of this training run

ANSWER: 1, 4. We haven't begun to overfit, so we can't know if we could do better by training longer or by increasing the capacity of the model, so those are things to try. Modifying learning rate or batch size is not a good idea at this stage because our model is learning but not yet overfitting. Collecting more data before overfitting to our current data may also not be a good idea.

(e) In one of his subsequent attempts at training the model, Dr. Pooler observed the following learning curve. Which additional steps should Dr. Pooler consider? Choose all that apply.



☐ Train for more epochs

☐ Use early stopping

☐ Apply dropout

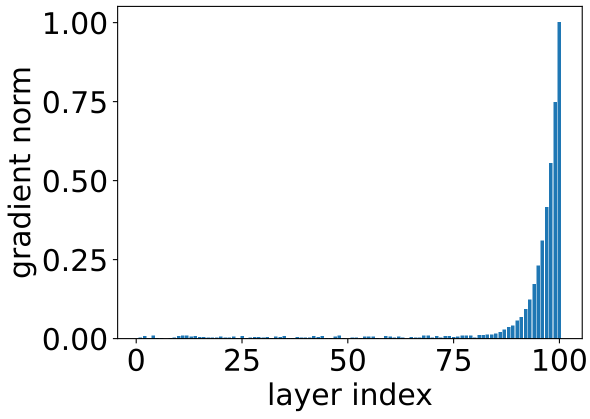
☐ Tune the hyperparameters of the model

☐ Collect more data

☐ Stop here and use the model obtained at the end of this training run

ANSWER: 2, 3, 4, 5. Once we've overfit, we want to find ways to improve generalization.

(f) Dr. Ada Grad, another colleague from ShallowBrain, suggested that it may be a good idea to train a much deeper network. Dr. Pooler trained a network with 100 layers and obtained the following plot showing the magnitudes of the gradients at each layer of the network. Describe the phenomenon that is being exhibited by the network and suggest potential solutions to address it.



ANSWER: The plot of the gradient norms demonstrates that the training is suffering from vanishing gradients. This is because we are training a very deep network. We studied two techniques that can help deal with vanishing gradients: batchnorm and skip connections.

Name: ____________________

MDP

2. (22 points) Consider a Markov decision process with four states $\{0, 1, 2, 3\}$ and two actions, $\{\text{left}, \text{right}\}$. The reward for any action in state 0 is 1, for any action in state 3 is 5, and for all other state action pairs is 0.

   (a) Assume that the transition function is deterministic such that taking action left in a state decreases the state by 1, taking action right increases the state by 1. Action left from state 0 does not change the state, and action right from state 3 does not change the state. Use $\gamma = 0.9$.

   i. Calculate the state-action value of horizon 1 $Q^1(s, a)$ for all (s, a) pairs.

$Q^1(s, a)$

	left	right
0		
1		
2		
3		

ANSWER:

$Q^1(s, a)$

	left	right
0	1	1
1	0	0
2	0	0
3	5	5

ii. Calculate the state-action value of horizon 2 $Q^2(s,a)$ for all (s,a) pairs.

$Q^2(s,a)$

	left	right
0		
1		
2		
3		

ANSWER:

$Q^2(s,a)$

	left	right
0	1.9	1
1	0.9	0
2	0	4.5
3	5	9.5

iii. What is the optimal horizon-2 policy for state 1? Your answer should be an *action*.

$\pi^{*2}(1) =$

ANSWER: $\pi^{*2}(1) =$ left

iv. Calculate the state-action value of horizon 3 $Q^3(s,a)$ for all (s,a) pairs.

Name: ______________________________

$Q^3(s, a)$

	left	right
0		
1		
2		
3		

ANSWER:

$Q^3(s, a)$

	left	right
0	2.71	1.81
1	1.71	4.05
2	0.81	8.55
3	9.05	13.55

v. What is the optimal horizon-3 policy for state 1? Your answer should be an *action*. Is there a change with respect to the horizon-2 policy? If so, explain why.

$\pi^{*3}(1) =$
The policy changed/did not change because...

ANSWER: $\pi^{*3}(1) =$ right. The policy changed because the agent now has sufficient time to reach the higher reward from state 3.

(b) Again assume the same deterministic transitions as in part (a). Using a value of $\gamma = 0.2$, what is the optimal horizon-3 policy for state 1? Show all your work. Is there a change with respect to the horizon-3 policy using $\gamma = 0.9$? If so, explain why.

ANSWER: $\pi^{*3}(1) = \text{right}$. We know that Q^1 does not change because it does not depend on the value of γ. Being clever, we could realize that we only need Q^2 for states 0 and 2, because $Q^3(1, \cdot)$ depends on $R(1, \cdot)$ the Q^2 values for states 0 and 2. So we have:

$Q^2(s, a)$

	left	right
0	1.2	1
1	...	...
2	0	1
3	...	...

Similarly, we only need Q^3 for state 1, which is all we need to compute the optimal policy for state 1: $Q^3(1, \text{left}) = 0.24$ and $Q(1, \text{right}) = 0.2$. With that, the optimal policy is $\pi^{*3}(1) = \text{left}$.
The policy changed from when $\gamma = 0.9$ because now future rewards are discounted more heavily. This makes the agent prefer the nearly-immediate, but smaller, reward of going left compared to the more-delayed, but larger, reward of going right.

(c) Find the largest value of γ such that the optimal horizon-3 policy for state 1 is to go left.

Name: ________________________________

ANSWER: $\gamma \leq 0.25$

$$Q^3(1,\text{left}) = 0 + \gamma(1 + \gamma(1))$$
$$Q^3(1,\text{right}) = 0 + \gamma(0 + \gamma(5)) Q^3(1,\text{left}) \geq Q^3(1,\text{right} \Rightarrow \qquad \gamma + \gamma^2 \geq 5\gamma^2$$
$$1 + \gamma \geq 5\gamma$$
$$4\gamma \leq 1$$
$$\gamma \leq 0.25$$

Name: ______________________________

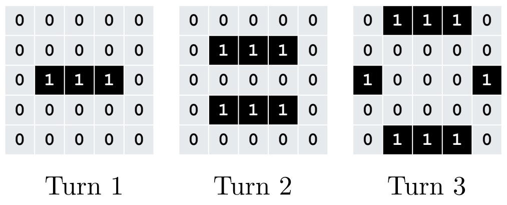
CNN Game of Life

3. (22 points) The Game of Life is a zero-player game devised by the British mathematician John Horton Conway in 1970. The game is played on an infinite black-and-white image. If a pixel is “alive”, it has a value of 1 (shown in black in images below). If the pixel is “dead”, it has a value of 0.

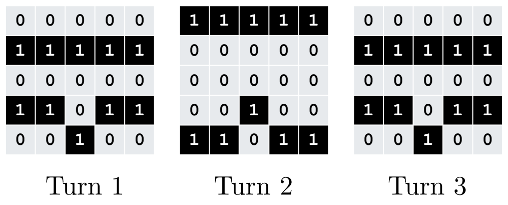
We will consider a variant of the game with the following rules: A player defines an initial image of alive and dead pixels. Then, every pixel “interacts” with the 8 pixels adjacent to it (above, below, right, left, and the four diagonals) and the following rules determine the image of the next turn:

1. Any live pixel with $\neq 3$ live neighbors dies (value becomes 0).
2. Any live pixel with exactly 3 live neighbors lives (value stays 1).
3. Any dead pixel with 2 or 3 live neighbors becomes a live pixel (value becomes 1).

Here are two examples:



Example 1



Example 2

In this problem, you will define the weights of a convolutional neural network (CNN) to implement the game rules. We use the convention `[width, height, channel_out, channel_in]` to define the dimensions of the filters below. Our neural network has the following architecture:

- Conv. layer with $3 \times 3 \times 2 \times 1$ filter weights $W^{(1)}$, s_1 stride, size p_1 padding with zeros, and no bias
- Conv. layer with $1 \times 1 \times 4 \times 2$ filter weights $W^{(2)}$, s_2 stride, size p_2 padding with zeros, and bias $b^{(2)}$ shape [4]
- Sigmoid activation
- Conv. layer with $1 \times 1 \times 2 \times 4$ filter weights $W^{(3)}$, s_2 stride, size p_2 padding with zeros, and bias $b^{(3)}$ shape [2]
- Sigmoid activation
- Conv. layer with $1 \times 1 \times 1 \times 2$ filter weights $W^{(4)}$, s_2 stride, size p_2 padding with zeros, and bias $b^{(4)}$ shape [1]
- Sigmoid activation ,

where s_1, s_2, p_1, and p_2 indicate values to be determined by you. The input is $x^{(1)}$ and output $a^{(4)}$. Note that, per the convention we have followed in class, padding of size p is applied to the left, right, top, and bottom of the input image.

(a) The first layer of our network will be a convolutional layer with two 3×3 filters that act on the single input channel.

i. Write down filter weights $W^{(1)}_{\text{channel\_out}=1,\ \text{channel\_in}=1}$ (for the 1st input channel and the 1st output channel) made of 1s and 0s that **counts the number of alive neighbors**.

ANSWER:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

.

ii. Write down the filter weights $W^{(1)}_{\text{channel\_out=2, channel\_in=1}}$ (for the 2nd output channel) made of 1s and 0s that can **determine whether a pixel is alive (1) or dead (0)**.

ANSWER:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

.

iii. What stride s_1 and size p_1 of padding with zeros should be used for the first convolutional layer? Explain your reasoning.

ANSWER: Because we want to output a prediction per pixel, and we are using 3×3 filters, we should use zero-padding of size 1 and stride of 1.

iv. Write down the numerical values of the two resulting images $z^{(1)}_{\text{channel=1}}$ and $z^{(1)}_{\text{channel}} = 2$ from applying your filters ($W^{(1)}_{\text{channel\_out=1, channel\_in=1}}$ and $W^{(1)}_{\text{channel\_out=2, channel\_in=1}}$, respectively) to Turn 1 of Example 2. Your answer should be two 5×5 grids of integers.

Name: ______________________________

ANSWER:

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

0	0	0	0	0
1	1	1	1	1
0	0	0	0	0
1	1	0	1	1
0	0	1	0	0

(b) We will use the second layer to determine whether the following conditions are true or false.

1. The number of live neighbors is ≤ 3
2. The number of live neighbors is ≥ 3
3. The number of live neighbors is ≥ 2
4. The pixel is alive

Following the convention from part (a), $z^{(1)}_{\text{channel}=1}$ is the value of the number of neighbors that we computed using our first convolutional filter above and $z^{(2)}_{\text{channel}=2}$ is the value for whether the pixel is alive or dead computed using our second convolutional filter above.

i. What stride s_2 and amount of padding with zero values p_2 should be used for the second – fourth convolutional layers? Explain your reasoning.

ANSWER: Because we want to output a prediction for each pixel, and we are using 1×1 filters, we should use no zero-padding ($p_2 = 0$) and a stride of 1.

Name: ________________________________

ii. $a^{(2)}$ is the output of the first sigmoid layer following our second convolution that should give the values of the conditions in the order given above (e.g., Condition 1 = Channel 1). Write what the outputs $a^{(2)}$ should be. Your answer should be four 5×5 grids of 0s and 1s.

ANSWER:

1	1	1	1	1
1	1	1	1	1
0	0	0	0	0
1	1	1	1	1
1	1	1	1	1

0	1	1	1	0
0	0	0	0	0
1	1	1	1	1
0	0	1	0	0
0	1	0	1	0

1	1	1	1	1
0	1	1	1	0
1	1	1	1	1
0	1	1	1	0
1	1	1	1	1

0	0	0	0	0
1	1	1	1	1
0	0	0	0	0
1	1	0	1	1
0	0	1	0	0

iii. Provide values for the filter weights $W^{(2)}$ of height and width = 1 and bias term $b^{(2)}$ that act on the outputs of the first convolutional layer $z^{(1)}$ to compute the values $z^{(2)}$ that we will pass through a sigmoid function to determine $a^{(2)}$, indicating whether conditions 1–4 (in that order) are true (1) or false (0). Each answer below should be a scalar value. *You may find it helpful to remind yourself at what values sigmoid approaches close to 1 or 0.*

$W^{(2)}_{\text{channel\_out=1, channel\_in=1}} =$ $W^{(2)}_{\text{channel\_out=1, channel\_in=2}} =$
$b^{(2)}_{\text{channel\_out=1}} =$

$W^{(2)}_{\text{channel\_out=2, channel\_in=1}} =$ $W^{(2)}_{\text{channel\_out=2, channel\_in=2}} =$
$b^{(2)}_{\text{channel\_out=2}} =$

$W^{(2)}_{\text{channel\_out=3, channel\_in=1}} =$ $W^{(2)}_{\text{channel\_out=3, channel\_in=2}} =$
$b^{(2)}_{\text{channel\_out=3}} =$

$W^{(2)}_{\text{channel\_out=4, channel\_in=1}} =$ $W^{(2)}_{\text{channel\_out=4, channel\_in=2}} =$
$b^{(2)}_{\text{channel\_out=4}} =$

ANSWER:

$$W^{(2)} = 15 \cdot \begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad b^{(2)} = 15 \cdot \begin{bmatrix} 3.5 & -2.5 & -1.5 & -0.5 \end{bmatrix}$$

Any reasonable large integer value could have been used in place of 15, so as to ensure that the output of the sigmoid is close to 0 or 1.

Name: ______________________________

(c) The game rules can be implemented in terms of our activated values of $a^{(2)}_{\text{channel=1}}$, $a^{(2)}_{\text{channel=2}}$, $a^{(2)}_{\text{channel=3}}$, and $a^{(2)}_{\text{channel=4}}$ as follows:

1. If alive with 3 neighbors, output True: $a^{(2)}_{\text{channel=1}}$ AND $a^{(2)}_{\text{channel=2}}$ AND $a^{(2)}_{\text{channel=4}}$
2. If dead with 2 or 3 neighbors, output True: $a^{(2)}_{\text{channel=1}}$ AND $a^{(2)}_{\text{channel=3}}$ AND NOT $a^{(2)}_{\text{channel=4}}$

Provide values for the filter weights $W^{(3)}$ with height and width $= 1$ and bias term $b^{(3)}$ that act on the outputs of the second convolutuonal layer and sigmoid $a^{(2)}$ to compute values $z^{(3)}$ that we will pass through a sigmoid to determine $a^{(3)}$, indicating whether conditions 1–2 (in that order) are true (1) or false (0). Each answer below should be a scalar value.

$W^{(3)}_{\text{channel\_out=1, channel\_in=1}} =$ $\qquad$ $W^{(3)}_{\text{channel\_out=1, channel\_in=2}} =$
$W^{(3)}_{\text{channel\_out=1, channel\_in=3}} =$ $\qquad$ $W^{(3)}_{\text{channel\_out=1, channel\_in=4}} =$
$b^{(3)}_{\text{channel\_out=1}} =$

$W^{(3)}_{\text{channel\_out=2, channel\_in=1}} =$ $\qquad$ $W^{(3)}_{\text{channel\_out=2, channel\_in=2}} =$
$W^{(3)}_{\text{channel\_out=2, channel\_in=3}} =$ $\qquad$ $W^{(3)}_{\text{channel\_out=2, channel\_in=4}} =$
$b^{(3)}_{\text{channel\_out=2}} =$

ANSWER:

$$W^{(3)} = 15 \cdot \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & -1 \end{bmatrix} \qquad b^{(2)} = 15 \cdot \begin{bmatrix} -2.5 & -1.5 \end{bmatrix}$$

Any reasonable large integer value could have been used in place of 15, so as to ensure that the output of the sigmoid is close to 0 or 1.

(d) Finally, for our last layer, we simply need to check whether either of the two conditions above are true. Provide values for the filter weights $W^{(4)}$ of height and width $= 1$ and bias term $b^{(4)}$ that act in the outputs of the third convolutional layer and sigmoid $a^{(3)}$ to compute the values $z^{(4)}$ that we wil pass through a sigmoid function to determine $a^{(4)}$, indicating whether a pixel is alive or dead in the next turn. Each answer below should be a scalar value.

$W^{(4)}_{\text{channel\_out=1, channel\_in=1}} =$ $\qquad$ $W^{(4)}_{\text{channel\_out=1, channel\_in=2}} =$
$b^{(4)}_{\text{channel\_out=1}} =$

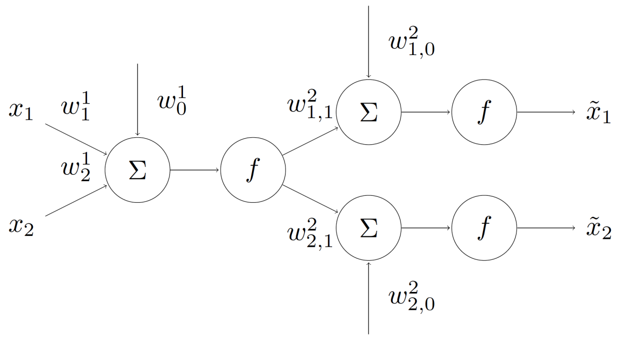
ANSWER:

$$W^{(4)} = 15 \cdot \begin{bmatrix} 1 & 1 \end{bmatrix} \qquad b^{(2)} = 15 \cdot \begin{bmatrix} -0.5 \end{bmatrix}$$

Any reasonable large integer value could have been used in place of 15, so as to ensure that the output of the sigmoid is close to 0 or 1.

Autencoders

4. (14 points) Otto N. Coder wants to find a way to represent his 2-dimensional data points in 1-dimensional space. Consider the following *autoencoder* with input $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathbb{R}^2$ and output $\tilde{x} = \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix} \in \mathbb{R}^2$. The autoencoder has one hidden layer with one hidden unit.



The encoder of this autoencoder is described by the output of the first hidden layer,

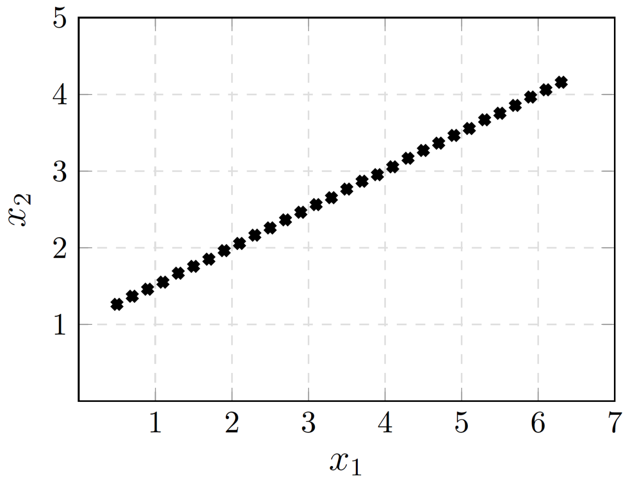
$$a^1 = f_1(w_1^1 x_1 + w_2^1 x_2 + w_0^1) \ ,$$

and the decoder of this autoencoder is described by the outputs of the output layer,

$$\tilde{x}_1 = f_2(w_{1,1}^2 a^1 + w_{1,0}^2) \ , \qquad \tilde{x}_2 = f_2(w_{2,1}^2 a_1 + w_{2,0}^2) \ .$$

The goal is to learn a set of weights such that $\tilde{x}_1 \approx x_1$ and $\tilde{x}_2 \approx x_2$ by means of first representing the input in a lower-dimensional space.

(a) For this part, suppose that the activation functions f_1, f_2 are the identity $f(z) = z$. Let our dataset $\mathcal{D}_n = \{x^{(i)}\}_{i=1}^n$ be composed of n data points, plotted below:



i. From the plot above, write a formula for x_2 as a function of x_1.

ANSWER: The data points roughly follow the equation $x_2 = 0.5x_1 + 1$.
Note that the autoencoder is NOT being used to solve a regression problem.

ii. Consider the autoencoder depicted in the figure at the start of this question. Using the result from the previous part, are there weights and biases for this autoencoder that perfectly recover $\tilde{x}_1 = x_1$ and $\tilde{x}_2 = x_2$, even though there is only one hidden unit? Find the weights and biases or justify why it is not possible.

$w_0^1 =$	$w_1^1 =$	$w_2^1 =$
$w_{1,0}^2 =$	$w_{1,1}^2 =$	
$w_{1,0}^2 =$	$w_{1,1}^2 =$	

Or justification:

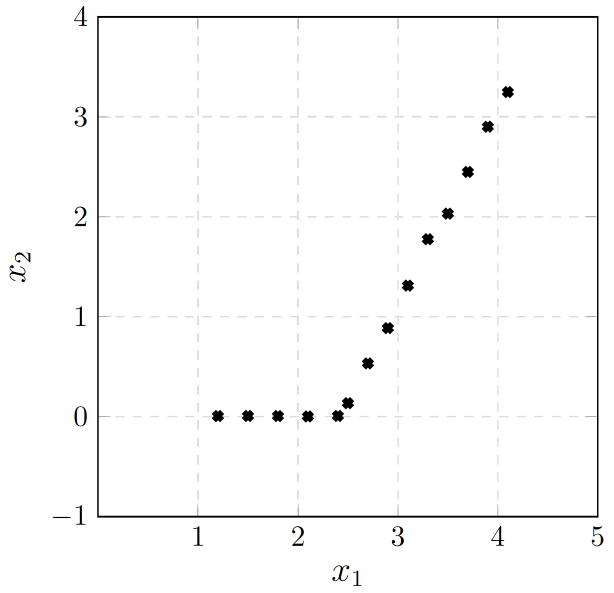
ANSWER:

$w_0^1 = 0$	$w_1^1 = 1$	$w_2^1 = 0$
$w_{1,0}^2 = 0$	$w_{1,1}^2 = 1$	
$w_{2,0}^2 = 1$	$w_{2,1}^2 = 0.5$	

iii. Briefly describe what these weights and biases in the autoencoder would be learning for this specific dataset.

ANSWER: The encoder would learn that x_2 is redundant with x_1, so it just encodes x_1 as $a^1 = x_1$ and discards x_2. Then the decoder would learn to reconstruct $\tilde{x}_1$ trivially as just $\tilde{x}_1 = a^1$ and (with slightly more work) reconstruct $\tilde{x}_2$ using the linear relationship from the previous part as $\tilde{x}_2 = 0.5a^1 + 1$.

(b) Otto modifies his autoencoder to have ReLU activation functions so that he can model nonlinear relationships, $f_1(z) = f_2(z) = \max(0, z)$. Suppose that he has a new dataset $\mathcal{D}_n = \{x^{(i)}\}_{i=1}^n$ composed of n data points, plotted below:



i. Again consider the autoencoder in the figure at the start of this question. Find values of the weights and biases in the autoencoder that perfectly recover this new dataset or justify why it is not possible.

$w_0^1 =$	$w_1^1 =$	$w_2^1 =$
$w_{1,0}^2 =$	$w_{1,1}^2 =$	
$w_{1,0}^2 =$	$w_{1,1}^2 =$	

Or justification:

ANSWER:

$w_0^1 = 0$	$w_1^1 = 1$	$w_2^1 = 0$
$w_{1,0}^2 = 0$	$w_{1,1}^2 = 1$	
$w_{2,0}^2 = -5$	$w_{2,1}^2 = 2$	

ii. Briefly describe what these weights and biases in the autoencoder would be learning for this specific dataset.

ANSWER: The encoder just encodes $a^1 = \max(0, x_1)$, which is almost the same as x_1 except that now all negative values of x_1 will be encoded into 0. From that encoded value, $\tilde{x}_2$ can still be perfectly reconstructed as $\tilde{x}_2 = \max(0, 2a^1 - 5) = \max(0, 2\max(0, x_1) - 5)$.

Name: ______________________________

Custom Sequence Generator

5. (11 points) ESE 577 alum Pat Tern proposes a novel approach to using attention mechanisms for generating custom mathematical sequences. Answer the following questions to understand how Pat's idea works. Pat begins with the standard self-attention code with autoregressive masking:

```
1)   def self_attention(x, d_k=4):
2)         """
3)         x has shape (n, d)               -- each row is an example
4)         """
5)         Wk = torch.nn.Linear(d, d_k, bias=False)
6)         Wq = torch.nn.Linear(d, d_k, bias=False)
7)         Wv = torch.nn.Linear(d, d_k, bias=False)
8)         k = Wk(x)
9)         q = Wq(x)
10)        M = torch.tril(torch.ones(n, n)) # return lower tri. & diag, others=0
11)        alpha = q @ k.transpose(-2,-1) * d_k**-0.5
12)        alpha = alpha.masked_fill(M == 0, float('-inf'))
13)        alpha = torch.softmax(alpha, dim=-1)
14)        v = Wv(x)
15)        out = alpha @ v
16)        return out
```

(a) Studying this core code for a self-attention head, Pat focuses on the attention weights α_{ij} computed in lines 11–13, and the embedding transform performed in line 14 by the network `Wv` defined in line 7. Answer the following questions for `x` with shape `(11, 3)` and `d_k=4`:

i. What is the shape of `alpha`?

ANSWER: $n \times n = 11 \times 11$.

ii. How many total free parameters are in the networks that the self-attention mechanism uses?

ANSWER: $3 \times d_k \times d = 36$.

(b) Pat modifies the self-attention code above to compute custom sequences where each term depends on prior terms based on a specific rule. She modifies the code above to demonstrate this for a machine that takes as input an $n = 3, d = 1$ sequence and computes `x[2] = (2 x[1] + x[0])/3`, using an attention matrix `alpha` with fixed values. This new procedure `custom_atention(...)` only uses one weight matrix, for values `v`:

```
1)  def custom_attention(x):
2)          Wv = torch.nn.Linear(d, d, bias=False)
3)          alpha = torch.tensor( ALPHA_MAT )
4)          alpha = torch.softmax(alpha, dim=-1)
5)          v = Wv(x)
6)          out = alpha @ v
7)          return out
```

Pat shows that when `x = torch.tensor([[a, b, c]]).T` is the input, then the output is `out = torch.tensor([[a, b, (2a + b) / 3]]).T`, for any integers a, b, and c.

i. For Pat to be correct, what should be the value of the `alpha` matrix computed in line 4, such that after training, the desired output is always obtained?

$$\texttt{alpha} = \begin{bmatrix} \quad & \quad & \quad \\ \quad & \quad & \quad \\ \quad & \quad & \quad \end{bmatrix}$$

ANSWER:

$$\texttt{alpha} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 2/3 & 1/3 & 0 \end{bmatrix}$$

ii. What should go into line 3 as `ALPHA_MAT`? (Hint: Recall that the softmax is defined as $\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$.)

$$\texttt{ALPHA\_MAT} = \begin{bmatrix} \quad & \quad & \quad \\ \\ \\ \end{bmatrix}$$

ANSWER:

$$\texttt{ALPHA\_MAT} = \begin{bmatrix} 1 & -\infty & -\infty \\ -\infty & 1 & -\infty \\ 1 + \log 2 & 1 & -\infty \end{bmatrix}$$

Any constant could replace the 1's. There must be an additive $\log 2$ in the bottom left entry to induce the $2 \times a$.

iii. What should be the learned values of the parameters of the network `Wv`?

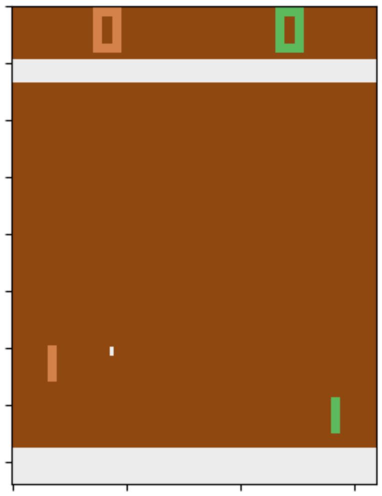
$$\texttt{Wv} =$$

ANSWER:

$$\texttt{Wv} = 1$$

RL for Pong

6. (19 points) We will use RL to train a policy to play the game of Pong in Atari. In Pong, the player controls a paddle by moving vertically up or down on the left side of the screen, and the opponent controls a similar paddle on the right side of the screen. A small ball moves left-to-right on the screen. If it hits the opponent's paddle, it bounces to the left and up or down depending on the angle of incidence; otherwise, the ball exits the screen on the rightmost end and the player gets one point. Similarly, when the ball moves right-to-left, it may hit the player's paddle or increase the opponent's score by one. The game ends when either player reaches 21 points. If the player wins, it obtains a reward of $+1$, and if it loses it obtains a reward of -1. There are three possible actions: stay, move up, move down. The figure below contains a screenshot of the game.



(a) Suppose that we discretize the state in the following way. There are 10 possible vertical locations for each of the player and the opponent, 30 possible vertical positions and 10 horizontal positions for the ball, 6 possible vertical velocities for the ball (including negative and positive values), and 6 possible horizontal velocities for the ball (also including negative and positive values). If we use Q-learning to train a policy to play the game, how many entries will the algorithm need to learn for the Q-table? (Note: Your answer should be a number, but a correct arithmetic expression is better than an incorrect number.)

ANSWER: $3 \times 10 \times 10 \times 30 \times 10 \times 6 \times 6 = 3,240,000$

Name: ________________________________

(b) Suppose instead that we use an image to represent the state and train a CNN as our Q-network. We will use grayscale images of size 84×84 and use the following architecture:

- Conv. layer with 16 filters of size 8×8, using a stride of 4 and no padding, followed by ReLU activation
- Conv. layer with 32 filters of size 4×4, using a stride of 2 and no padding, followed by ReLU activation.
- Flatten
- Fully-connected layer with 256 neurons, followed by ReLU activation
- Fully-connected layer to compute the desired Q-values
- Unknown output activation

i. What is the size of the output of the second conv. layer?

ANSWER: $9 \times 9 \times 32$. For the first layer, we obtain $H = W = \lceil \frac{84-7}{4} \rceil = 20$. Then for the second layer, we obtain $H = W = \lceil \frac{20-3}{2} \rceil = 9$

ii. What is the number of neurons in the output layer of the network?

ANSWER: 3. This is equal to the number of actions.

iii. What activation function should we use for the final layer?

ANSWER: Linear activation. Note that ReLU is not appropriate because there are negative rewards (and therefore possibly negative Q-values).

iv. What is the total number of learnable parameters in the Q-network? (Note: Your answer should be a number, but a correct arithmetic expression is better than an incorrect number.)

ANSWER: $(8 \cdot 8 \cdot 16) + (4 \cdot 4 \cdot 16 \cdot 32) + (9 \cdot 9 \cdot 32 \cdot 256) + (256 \cdot 3) = 673,536$. As is common practice in deep learning, many numbers here are powers of two. Factoring out 2^8 was one way to somewhat quickly obtain this number. Note: since it wasn't specified whether the networks used biases or not, either assumption was treated as correct (the calculations above assume no bias).

(c) Unfortunately, representing the state of the game as an image is insufficient for even this very simple game. Explain why. (Hint: think about yourself deciding how to move the player's paddle upon observing one screenshot of the game.)

ANSWER: A single image does not contain velocity information, so we cannot know which direction the ball is moving or which direction the opponent's paddle is moving, or how fast.

(d) One possible alternative is to use multiple consecutive images as the state representation. Suppose we choose to process 4 consecutive screenshots instead of just one. How could we modify our CNN to achieve this? How many learnable parameters would the new CNN have?

ANSWER: We should stack the four consecutive frames into a $84 \times 84 \times 4$ image and pass that as the input to our network. This would result in adding $8 \cdot 8 \cdot 16 \cdot 3 = 3,072$ to our original count, yielding 676,608. Note: since it wasn't specified whether the networks used biases or not, either assumption was treated as correct (the calculations above assume no bias).