Question 1 Multilayer Perceptron.
Give the weights and biases of a multilayer perceptron which takes as input two scalar values (x_1, x_2) and outputs the values in sorted order. The hidden units should all use the ReLU activation function, and the output units should be linear. You should explain why your solution works, but you don't need to provide a formal proof.

The multilayer perceptron after removing the lines with a weight of 0 is shown in Figure 1.

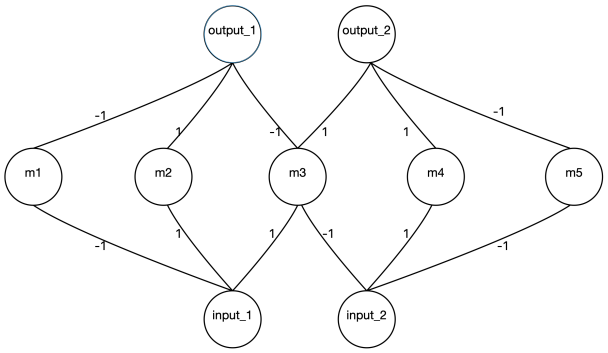

Figure 1: The multilayer perceptron without 0-weight lines.

$input_1$ and $input_2$ is the input of two values. $output_1$ is the less number of the input values and $output_2$ is the greater one. m_1 to m_5 shows the hidden units with ReLU as their activation function. All biases are 0.

For simple cases, where $input_1$ and $input_2$ are both greater than or equal to 0, m_1 and m_5 will both be 0 and m_2 is a copy of $input_1$; similarly, m_4 is a copy of $input_2$. m_3 calculates $(input_1 - input_2)$. When $input_1 \geq input_2$, m_3 will be a positive number, $(input_1 - input_2)$. Then, $output_1$ will be $input_1 - (input_1 - input_2) = input_2$, which is the less number. $output_2$ will be $input_2 + (input_1 - input_2) = input_1$. When $input_1 < input_2$, m_3 will become 0. Then, $output_1$ will be $input_1$, which is the less number, and $output_2$ will be $input_2$.

For the cases where $input_1$ and $input_2$ can be negative numbers. The role of m_1 is to store the value of $input_1$ instead of m_2 when $input_1$ is less than 0, where m_2 will be 0 according to the ReLU activation function.

The complete multilayer perceptron is shown in Figure 2.

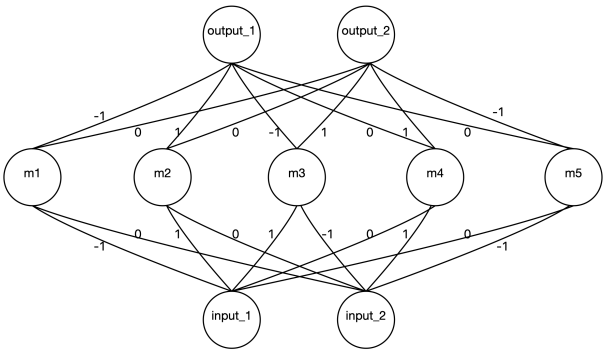

Figure 2: The multilayer perceptron for sorting input values. The hidden units use the ReLU activation function, and the output units are linear.

Question 2 Backprop.

(a) *Draw the computation graph for all the variables.*

The computation graph is shown in Figure 3.

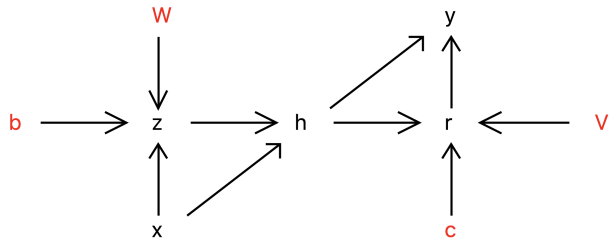

Figure 3: The computation graph for all the variables.

(b) Determine the backprop rules (in vector form) for computing the gradients with respect to all the parameters.

Forward pass:

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$
$$\mathbf{h} = \phi(\mathbf{z}) + \mathbf{x}$$
$$\mathbf{r} = \mathbf{V}\mathbf{h} + \mathbf{c}$$
$$\mathbf{y} = \phi(\mathbf{r}) + \mathbf{h}$$

Backward pass:

$$\overline{\mathbf{y}} = \mathbf{1}$$
$$\overline{\mathbf{r}} = \overline{\mathbf{y}} \circ \frac{\partial \mathbf{y}}{\partial \mathbf{r}} = \phi'(\mathbf{r})$$
$$\frac{\partial \mathbf{r}}{\partial \mathbf{V}} = \mathbf{1}\mathbf{h}^T$$
$$\frac{\partial \mathbf{r}}{\partial \mathbf{c}} = \mathbf{1}$$
$$\overline{\mathbf{V}} = \overline{\mathbf{r}}(\frac{\partial \mathbf{r}}{\partial \mathbf{V}})^T = \phi'(\mathbf{r})\mathbf{h}^T$$
$$\overline{\mathbf{c}} = \overline{\mathbf{r}} \circ \frac{\partial \mathbf{r}}{\partial \mathbf{c}} = \phi'(\mathbf{r})$$
$$\overline{\mathbf{h}} = \overline{\mathbf{y}} \circ \frac{\partial \mathbf{y}}{\partial \mathbf{h}} + \overline{\mathbf{r}} \circ \frac{\partial \mathbf{r}}{\partial \mathbf{h}} = \mathbf{1} + \mathbf{V}^T\phi'(\mathbf{r})$$
$$\overline{\mathbf{z}} = \overline{\mathbf{h}} \circ \frac{\partial \mathbf{h}}{\partial \mathbf{z}} = \overline{\mathbf{h}} \circ \phi'(\mathbf{z}) = \phi'(\mathbf{z}) + \mathbf{V}^T\phi'(\mathbf{r}) \circ \phi'(\mathbf{z})$$
$$\overline{\mathbf{W}} = \overline{\mathbf{z}}(\frac{\partial \mathbf{z}}{\partial \mathbf{W}})^T = [\phi'(\mathbf{z}) + \mathbf{V}^T\phi'(\mathbf{r}) \circ \phi'(\mathbf{z})]\mathbf{x}^T$$
$$\overline{\mathbf{b}} = \overline{\mathbf{z}} \circ \frac{\partial \mathbf{z}}{\partial \mathbf{b}} = [\phi'(\mathbf{z}) + \mathbf{V}^T\phi'(\mathbf{r}) \circ \phi'(\mathbf{z})] \circ \mathbf{1}$$
$$= \phi'(\mathbf{z}) + \mathbf{V}^T\phi'(\mathbf{r}) \circ \phi'(\mathbf{z})$$

$\mathbf{x}$, $\mathbf{y}$ and $\mathbf{h}$ all have the same size. $\mathbf{1}$ is a vector with the same size as $\mathbf{y}$, whose each unit is 1.

Question 3 AlexNet.

(a) *Count the number of units, the number of weights, and the number of connections in each layer.*

	# Units	# Weights (Parameters)	# Connections
Input	3*224*224=150,528		
Convolution Layer 1	96*55*55=290,400	11*11*3*96=34,848	11*11*3*96*55*55=105,415,200
Pooling Layer	96*27*27=69,984		
Convolution Layer 2	256*27*27=186,624	5*5*48*128*2=307,200	5*5*48*256*27*27=223,948,800
Pooling Layer	256*13*13=43,264		
Convolution Layer 3	384*13*13=64,896	3*3*256*384=884,736	3*3*256*384*13*13=149,520,384
Convolution Layer 4	384*13*13=64,896	3*3*192*192*2=663,552	3*3*129*384*13*13=112,140,288
Convolution Layer 5	256*13*13=43,264	3*3*192*128*2=442,368	3*3*192*256*13*13=74,760,192
Pooling Layer	256*6*6=9,216		
Fully Connected Layer 1	4,096	9216*4096=37,748,736	9216*4096=37,748,736
Fully Connected Layer 2	4,096	4096*4096=16,777,216	4096*4096=16,777,216
Output Layer	1,000	4096*1000=4,096,000	4096*1000=4,096,000

Layers 2, 4 & 5 are not connected to layers between GPUs; thus, we calculate the parameters and connections of each component and multiplied by 2.

(b) *For each of the following scenarios, based on your answers to Part 1, suggest a change to the architecture which will help achieve the desired objective. I.e., modify the sizes of one or more layers.*

i. *You want to reduce the memory usage at test time so that the network can be run on a cell phone; this requires reducing the number of parameters for the network.*

The percentages of the number of parameters of convolutional layers and connected layers are 3.8% and 96.2% respectively. Thus, a more efficient way to reduce the number of parameters for the network is to reduce the parameters in connected layers.
I will change the units of Fully Connected Layer 1 and 2 from 4096 to 2048; consequently, the number of parameters of the new network will become 56.5% of the original one.

ii. *Your network will need to make very rapid predictions at test time. You want to reduce the number of connections, since there is approximately one add-multiply operation per connection.*

The percentages of the number of connections (or computations) of convolutional layers and connected layers are 8.2% and 91.8% respectively. As a result, it is more efficient to reduce the number of connections of convolutional layers for making rapid predictions.
I will change the size of Convolution Layer 1 and 2 from $(55 \times 55 \times 96)$, $(27 \times 27 \times 256)$ to $(27 \times 27 \times 96)$, $(13 \times 13 \times 256)$. The number of computations will become 64.3% of the original.