Princeton University Computer Science
Fall 2025

COS 324: Introduction to Machine Learning
Backpropagation

Wuwei Zhang, Edin Hasanovic, Ruth Fong

Last updated: Monday 27th October, 2025, 12:31pm.

Note: These are new notes for Fall 2025 and may have some typos; please post on Ed if you notice any.

1. Overview of Training Feedforward Neural Networks

1.1. **Forward Pass Computation.** A feedforward neural network consists of multiple layers of transformations applied sequentially to an input vector $\vec{\mathbf{x}}$. Each layer applies a linear transformation followed by a non-linear activation function. For example, consider the following network with 2 hidden layers:

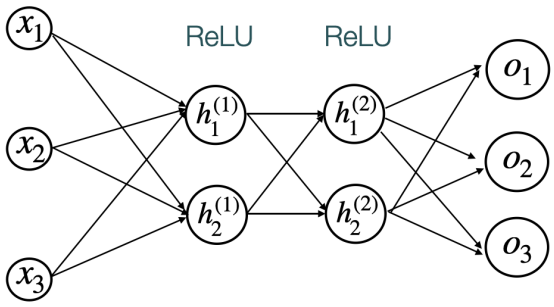

Figure 1. 2-layer network diagram.

$$\vec{\mathbf{h}}^{(1)} = \text{ReLU}(\mathbf{W}^{(1)}\vec{\mathbf{x}}), \quad \vec{\mathbf{h}}^{(2)} = \text{ReLU}(\mathbf{W}^{(2)}\vec{\mathbf{h}}^{(1)}), \quad \vec{\mathbf{o}} = \mathbf{W}^{(o)}\vec{\mathbf{h}}^{(2)} \tag{1}$$

For this network:

- $\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \mathbf{W}^{(o)}$ are the weight matrices for each layer.
- $\text{ReLU}(z) = \max(0, z)$ is the non-linear activation function.
- $\vec{\mathbf{o}}$ represents the raw output (Note: The raw output vector is sometimes passed through a softmax or sigmoid function in order to represent probability scores between 0 and 1).

See Course Notes (CN) Exercise 11.1.3 (link) for an example of computing the forward pass through a network given the network weights and an input vector.

1.2. **Training Objective.** The goal of training is to find weight parameters that minimize a loss function over the training dataset $\mathcal{D} = \{(\vec{\mathbf{x}}, y)\}$. Typically, for classification tasks, we minimize the cross-entropy loss L given in Equation (2):

$$\min_{\mathbf{W}^{(1)},\mathbf{W}^{(2)},\ldots,\mathbf{W}^{(L)}} L = \min_{\mathbf{W}^{(1)},\mathbf{W}^{(2)},\ldots,\mathbf{W}^{(L)}} \sum_{(\vec{\mathbf{x}},y)\in\mathcal{D}} -\log \widehat{o}_y \tag{2}$$

In the above loss:

- y is the true class label.
- $\widehat{o}_i = \dfrac{\exp(o_i)}{\sum_{k=1}^{K} \exp(o_k)}$ is the softmax score for class i.

Thus, $\widehat{o}_y$ is the softmax score for the true class label y. This optimization problem requires computing gradients of the loss with respect to all parameters in the network.

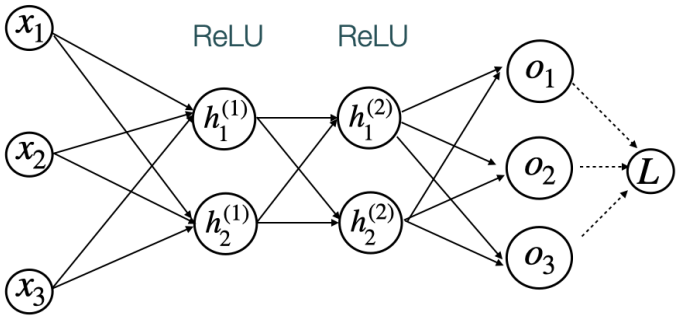

FIGURE 2. 2-layer network diagram with loss node L added.

1.3. **Gradient-Based Optimization.** Training neural networks typically proceeds via gradient descent or some variant (e.g. SGD); the general procedure is as follows:

1. Randomly initialize all weights $\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \ldots, \mathbf{W}^{(o)}$.
2. Compute the forward pass to obtain network outputs and the loss L.
3. Compute the gradients of the loss with respect to each parameter:
$$\frac{\partial L}{\partial \mathbf{W}^{(1)}}, \quad \frac{\partial L}{\partial \mathbf{W}^{(2)}}, \quad \frac{\partial L}{\partial \mathbf{W}^{(o)}}$$
4. Update parameters using the gradient descent rule:
$$\mathbf{W}^{(l)} \leftarrow \mathbf{W}^{(l)} - \eta \frac{\partial L}{\partial \mathbf{W}^{(l)}}$$
where η is the learning rate.

1.4. **Backpropagation: Computing the Gradients Efficiently.** The backpropagation algorithm allows us to efficiently compute the gradients of the loss with respect to (w.r.t.) each parameter (i.e. step 3 in Section 1.3) by using the chain rule from calculus and works as follows:

1. Compute $\dfrac{\partial L}{\partial \mathbf{W}^{(o)}}$, the gradient of the loss w.r.t. the output layer.[1]
2. Compute gradients one layer at a time, iterating backward from the last layer to the first and using the chain rule.

[1]We abuse the term gradient in these notes to keep things simple. Formally, if something of the form $\dfrac{\partial f}{\partial x}$ is a scalar, it is known as a partial derivative; if it is a vector or matrix (e.g. $\dfrac{\partial f}{\partial \vec{\mathbf{u}}}$, $\dfrac{\partial \vec{\mathbf{u}}}{\partial x}$, or $\dfrac{\partial \vec{\mathbf{u}}}{\partial \vec{\mathbf{v}}}$), it is known as a Jacobian matrix. See Course Notes (CN) 11.4.1 (link) for more on the Jacobian matrix.

Formally, for each layer l, the gradient of the loss w.r.t. its weight matrix is as follows:

$$\frac{\partial L}{\partial \mathbf{W}^{(l)}} = \frac{\partial L}{\partial \vec{\mathbf{h}}^{(l)}} \frac{\partial \vec{\mathbf{h}}^{(l)}}{\partial \mathbf{W}^{(l)}} \tag{3}$$

and the gradient of the loss w.r.t. the previous layer's activations is as follows:

$$\frac{\partial L}{\partial \vec{\mathbf{h}}^{(l-1)}} = \frac{\partial L}{\partial \vec{\mathbf{h}}^{(l)}} \frac{\partial \vec{\mathbf{h}}^{(l)}}{\partial \vec{\mathbf{h}}^{(l-1)}} \tag{4}$$

Since each layer depends only on the activations of the previous layer in a feedforward network, the computation can be performed efficiently in reverse order, thereby avoiding redundant calculations.

2. Backpropagation

To better understand how backpropagation operates, we will walk through a concrete example using the network shown in Figure 1. Our goal is to compute the gradients of the loss function with respect to each layer's parameters:

$$\frac{\partial L}{\partial \mathbf{W}^{(1)}}, \quad \frac{\partial L}{\partial \mathbf{W}^{(2)}}, \quad \frac{\partial L}{\partial \mathbf{W}^{(o)}}$$

We will do this step by step, illustrating how the backward pass proceeds from the output layer back to the input layer.

2.1. Overview.

Using the backpropagation algorithm to compute gradients for this network can be divided into the following four steps:

1. Compute $\frac{\partial L}{\partial o_i}$ for each output neuron o_i. This measures how the loss changes with respect to the model's raw output scores before applying the softmax.
2. Propagate gradients from the output layer to the last hidden layer:

$$\frac{\partial L}{\partial \vec{\mathbf{h}}^{(2)}} = \frac{\partial L}{\partial \vec{\mathbf{o}}} \cdot \frac{\partial \vec{\mathbf{o}}}{\partial \vec{\mathbf{h}}^{(2)}}$$

   and then compute $\frac{\partial L}{\partial \mathbf{W}^{(o)}}$.
3. Propagate further backward through earlier hidden layers (e.g., through ReLU activations and $\mathbf{W}^{(2)}$) to compute $\frac{\partial L}{\partial \vec{\mathbf{h}}^{(1)}}$ and $\frac{\partial L}{\partial \mathbf{W}^{(2)}}$.
4. Finally, compute gradients for the first layer's parameters: $\frac{\partial L}{\partial \mathbf{W}^{(1)}}$, completing the chain of derivatives from the loss back to the input.

Each step reuses intermediate results from the forward pass (such as ReLU activations and pre-softmax outputs), making the algorithm efficient.

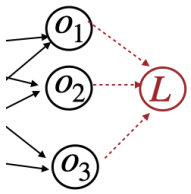

FIGURE 3. Output layer and loss layer.

2.2. **Step 1: $\frac{\partial L}{\partial o_i}$, i.e. Gradient of the Loss with Respect to Output Neuron o_i.** The first step in backpropagation begins at the end of the network—computing how the loss changes with respect to each output node.

Setup. Let the network output raw scores $\vec{\mathbf{o}} = [o_1, o_2, o_3]^\top$. We convert these into predicted probabilities using the softmax function:

$$\widehat{o}_i = \frac{\exp(o_i)}{\sum_{k=1}^{K} \exp(o_k)} \tag{5}$$

and define the loss for a single example (with true label y) using the cross-entropy objective:

$$L = -\log \widehat{o}_y = -\log \left(\frac{\exp(o_y)}{\sum_{k=1}^{K} \exp(o_k)} \right) \tag{6}$$

Derivative Derivation. Using properties of logarithms,

$$L = -o_y + \log \left(\sum_{k=1}^{K} \exp(o_k) \right) \tag{7}$$

Now, we compute the gradient with respect to each o_i and use the definition of the softmax function (Equation (5)):

- For the target class $i = y$:

$$\frac{\partial L}{\partial o_y} = -1 + \frac{\exp(o_y)}{\sum_{k=1}^{K} \exp(o_k)} = -1 + \widehat{o}_y \tag{8}$$

- For non-target classes $i \neq y$:

$$\frac{\partial L}{\partial o_i} = 0 + \frac{\exp(o_i)}{\sum_{k=1}^{K} \exp(o_k)} = \widehat{o}_i \tag{9}$$

Combining both cases:

$$\frac{\partial L}{\partial o_i} = \begin{cases} \widehat{o}_i - 1 & \text{if } i = y \\ \widehat{o}_i & \text{otherwise} \end{cases} \tag{10}$$

Indicator function. Before we continue, we'll define the indicator function $\mathbb{1}[c]$ for some conditional statement c:

$$\mathbb{1}[c] = \begin{cases} 1, & \text{if } c \text{ is true} \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

Now, Equation (10) can be written using the indicator function $\mathbb{1}[\cdot]$:

$$\frac{\partial L}{\partial o_i} = \widehat{o}_i - \mathbb{1}[i = y] \tag{12}$$

This is an extremely useful result—it shows that the gradient at the output layer is simply the difference between the predicted probability and the one-hot encoded ground truth label.

Interpretation.

- The gradient is negative for the correct class ($\widehat{o}_y - 1 < 0$), meaning the logit o_y (i.e. pre-softmax value) should *increase* in order to reduce the loss.
- The gradient is positive for all incorrect classes ($\widehat{o}_i > 0$), meaning the logits for all incorrect classes (i.e. o_i, where $i \neq y$) should *decrease*.

Example. Suppose we have $x_1 = x_2 = x_3 = 1$, and the true label is $y = 3$.

From the forward pass (using the given network parameters):

$$[o_1, o_2, o_3] = [1, 3, 2]$$

Then the softmax probabilities are:

$$\widehat{o}_i = \frac{\exp(o_i)}{\exp(1) + \exp(3) + \exp(2)}$$

Thus:

$$\frac{\partial L}{\partial o_1} = \widehat{o}_1 = \frac{e^1}{e^1 + e^3 + e^2}, \quad \frac{\partial L}{\partial o_2} = \widehat{o}_2 = \frac{e^3}{e^1 + e^3 + e^2}, \quad \frac{\partial L}{\partial o_3} = \widehat{o}_3 - 1 = \frac{e^2}{e^1 + e^3 + e^2} - 1$$

2.3. **Step 2: Gradients w.r.t. second hidden layer and output weights.** At this stage, we already know how the loss changes with respect to each output neuron:

$$\frac{\partial L}{\partial o_1}, \quad \frac{\partial L}{\partial o_2}, \quad \frac{\partial L}{\partial o_3}$$

Our next goal is to propagate these gradients backward to the previous layer by computing the following gradients:

$$\frac{\partial L}{\partial h_1^{(2)}}, \quad \frac{\partial L}{\partial h_2^{(2)}}, \quad \frac{\partial L}{\partial \mathbf{W}^{(o)}}$$

These gradients tell us the following:

- $\frac{\partial L}{\partial h_1^{(2)}}, \frac{\partial L}{\partial h_2^{(2)}}$: how the loss changes w.r.t. each activation in the last hidden layer.
- $\frac{\partial L}{\partial \mathbf{W}^{(o)}}$: how each weight connecting $\vec{\mathbf{h}}^{(2)}$ to the outputs should be updated.

This step is often split into two sub-steps:

- **Step 2A:** Compute gradient w.r.t. hidden units: $\frac{\partial L}{\partial \vec{\mathbf{h}}^{(2)}} = [\frac{\partial L}{\partial h_1^{(2)}}, \frac{\partial L}{\partial h_2^{(2)}}]^\top$
- **Step 2B:** Compute gradient w.r.t. weights: $\frac{\partial L}{\partial \mathbf{W}^{(o)}}$

Step 2A: Compute gradient w.r.t. hidden units: $\frac{\partial L}{\partial \vec{\mathbf{h}}^{(2)}}$.

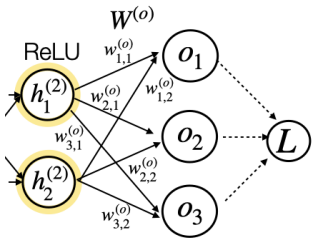

FIGURE 4. Partial network diagram highlighting hidden units in layer 2.

We now want to find how the loss L changes with respect to the hidden activations $h_1^{(2)}$ and $h_2^{(2)}$ (see Section 2.3). From the network definition (Equation (1)), we have the following:

$$o_i = \sum_j w_{i,j}^{(o)} h_j^{(2)} \tag{13}$$

Here, each output neuron o_i depends linearly on the hidden activations $h_1^{(2)}, h_2^{(2)}$ through the output-layer weights $w_{i,j}^{(o)}$.

Below, we explicitly write out how to compute o_1, o_2, o_3 in our example network:

$$\begin{aligned} o_1 &= w_{1,1}^{(o)} h_1^{(2)} + w_{1,2}^{(o)} h_2^{(2)} \\ o_2 &= w_{2,1}^{(o)} h_1^{(2)} + w_{2,2}^{(o)} h_2^{(2)} \\ o_3 &= w_{3,1}^{(o)} h_1^{(2)} + w_{3,2}^{(o)} h_2^{(2)} \end{aligned} \tag{14}$$

Thus, the partial derivatives are as follows:

$$\frac{\partial o_1}{\partial h_1^{(2)}} = w_{1,1}^{(o)}, \quad \frac{\partial o_2}{\partial h_1^{(2)}} = w_{2,1}^{(o)}, \quad \frac{\partial o_3}{\partial h_1^{(2)}} = w_{3,1}^{(o)} \tag{15}$$

Recall that we're interested in finding $\frac{\partial L}{\partial \vec{\mathbf{h}}^{(2)}}$; we'll focus on finding $\frac{\partial L}{\partial h_1^{(2)}}$ first (i.e. gradient of loss with respect to the first hidden unit in layer 2). Using the chain rule, we have the following:

$$\frac{\partial L}{\partial h_1^{(2)}} = \frac{\partial L}{\partial o_1}\frac{\partial o_1}{\partial h_1^{(2)}} + \frac{\partial L}{\partial o_2}\frac{\partial o_2}{\partial h_1^{(2)}} + \frac{\partial L}{\partial o_3}\frac{\partial o_3}{\partial h_1^{(2)}} \tag{16}$$

We can re-write Equation (16) using the following compact matrix notation:

$$\frac{\partial L}{\partial h_1^{(2)}} = \begin{bmatrix} \frac{\partial L}{\partial o_1} & \frac{\partial L}{\partial o_2} & \frac{\partial L}{\partial o_3} \end{bmatrix} \begin{bmatrix} w_{1,1}^{(o)} \\ w_{2,1}^{(o)} \\ w_{3,1}^{(o)} \end{bmatrix} \tag{17}$$

We can write a similar equation for $h_2^{(2)}$:

$$\frac{\partial L}{\partial h_2^{(2)}} = \begin{bmatrix} \frac{\partial L}{\partial o_1} & \frac{\partial L}{\partial o_2} & \frac{\partial L}{\partial o_3} \end{bmatrix} \begin{bmatrix} w_{1,2}^{(o)} \\ w_{2,2}^{(o)} \\ w_{3,2}^{(o)} \end{bmatrix} \tag{18}$$

Finally, we can stack these results together to give the following:

$$\frac{\partial L}{\partial \vec{\mathbf{h}}^{(2)}} = \left(\frac{\partial L}{\partial \vec{\mathbf{o}}}\right)^\top \mathbf{W}^{(o)}, \quad \text{where } \mathbf{W}^{(o)} = \begin{bmatrix} w_{1,1}^{(o)} & w_{1,2}^{(o)} \\ w_{2,1}^{(o)} & w_{2,2}^{(o)} \\ w_{3,1}^{(o)} & w_{3,2}^{(o)} \end{bmatrix} \tag{19}$$

Interpretation. Each hidden unit $h_j^{(2)}$ influences all output neurons o_i. The gradient $\frac{\partial L}{\partial h_j^{(2)}}$ therefore aggregates contributions from all outputs that depend on it, weighted by their respective connection strengths $w_{i,j}^{(o)}$.

Step 2B: Compute gradient w.r.t. to weights: $\frac{\partial L}{\partial \mathbf{W}^{(o)}}$.

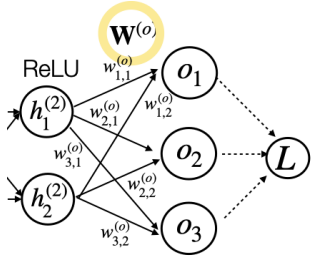

FIGURE 5. Partial network diagram highlighting output weights.

We will now find how the loss changes with respect to the weights connecting the hidden activations $\vec{\mathbf{h}}^{(2)}$ to the output layer $\vec{\mathbf{o}}$.

Let's focus on first computing $\frac{\partial L}{\partial w_{2,1}^{(o)}}$. Recall how we compute o_1, o_2, o_3 (Equation (14)):

$$\begin{aligned} o_1 &= w_{1,1}^{(o)} h_1^{(2)} + w_{1,2}^{(o)} h_2^{(2)} \\ o_2 &= w_{2,1}^{(o)} h_1^{(2)} + w_{2,2}^{(o)} h_2^{(2)} \\ o_3 &= w_{3,1}^{(o)} h_1^{(2)} + w_{3,2}^{(o)} h_2^{(2)} \end{aligned}$$

Here, we see that $w_{2,1}^{(o)}$ is only used in the computation for o_2 (not o_1 or o_3):

$$\frac{\partial o_1}{\partial w_{2,1}^{(o)}} = 0, \quad \frac{\partial o_2}{\partial w_{2,1}^{(o)}} = h_1^{(2)}, \quad \frac{\partial o_3}{\partial w_{2,1}^{(o)}} = 0 \tag{20}$$

Now, using the chain rule, we have the following:

$$\frac{\partial L}{\partial w_{2,1}^{(o)}} = \frac{\partial L}{\partial \vec{\mathbf{o}}} \frac{\partial \vec{\mathbf{o}}}{\partial w_{2,1}^{(o)}} = \begin{bmatrix} \frac{\partial L}{\partial o_1} & \frac{\partial L}{\partial o_2} & \frac{\partial L}{\partial o_3} \end{bmatrix} \begin{bmatrix} \frac{\partial o_1}{w_{2,1}^{(o)}} \\ \frac{\partial o_2}{w_{2,1}^{(o)}} \\ \frac{\partial o_3}{w_{2,1}^{(o)}} \end{bmatrix} \tag{21}$$

$$\frac{\partial L}{\partial w_{2,1}^{(o)}} = \frac{\partial L}{\partial o_1} \frac{\partial o_1}{\partial w_{2,1}^{(o)}} + \frac{\partial L}{\partial o_2} \frac{\partial o_2}{\partial w_{2,1}^{(o)}} + \frac{\partial L}{\partial o_3} \frac{\partial o_3}{\partial w_{2,1}^{(o)}} \tag{22}$$

Finally, substituting the partial derivatives from Equation (20) into Equation (22) gives us the following:

$$\frac{\partial L}{\partial w_{2,1}^{(o)}} = \frac{\partial L}{\partial o_2} \times h_1^{(2)} \tag{23}$$

Generalization. By following a similar process for all other output weights of the form $w_{i,j}^{(o)}$, we have the following:

$$\frac{\partial L}{\partial w_{i,j}^{(o)}} = \frac{\partial L}{\partial o_i} \times h_j^{(2)} \tag{24}$$

This captures the contribution of hidden activation $h_j^{(2)}$ to the output neuron o_i's error. When we collect all the partial derivatives of the form $\frac{\partial L}{\partial w_{i,j}^{(o)}}$ from Equation (24) into a matrix, we have the following:

$$\frac{\partial L}{\partial \mathbf{W}^{(o)}} = \begin{bmatrix} \frac{\partial L}{\partial o_1} h_1^{(2)} & \frac{\partial L}{\partial o_1} h_2^{(2)} \\ \frac{\partial L}{\partial o_2} h_1^{(2)} & \frac{\partial L}{\partial o_2} h_2^{(2)} \\ \frac{\partial L}{\partial o_3} h_1^{(2)} & \frac{\partial L}{\partial o_3} h_2^{(2)} \end{bmatrix} \tag{25}$$

We can write this compactly as follows:

$$\frac{\partial L}{\partial \mathbf{W}^{(o)}} = \left(\frac{\partial L}{\partial \vec{\mathbf{o}}} \right)^\top \left(\vec{\mathbf{h}}^{(2)} \right)^\top \tag{26}$$

Note: You may find it helpful to check matrix dimensions: $\left(\frac{\partial L}{\partial \vec{\mathbf{o}}} \right)^\top$ is a 3×1 matrix, $\left(\vec{\mathbf{h}}^{(2)} \right)^\top$ is a 1×2 matrix; thus, the product is the 3×2 matrix given in Equation (25).

Interpretation. Each partial derivative term in the matrix given by Equation (25) is the product of the following two terms:

- the **error signal** from the output neuron, $\frac{\partial L}{\partial o_i}$, and
- the **input activation** that the weight connected from, $h_j^{(2)}$.

Large errors and large activations both lead to larger weight updates, reflecting the idea that the weights contributing most to the error should be corrected the most.

Example. Given the same forward activations from the previous example:

$$\vec{\mathbf{h}}^{(2)} = [1,\, 2]^\top \quad \vec{\mathbf{o}} = [1,\, 3,\, 2]^\top$$

We already computed:

$$\frac{\partial L}{\partial \vec{\mathbf{o}}} = [\,\widehat{o}_1,\, \widehat{o}_2,\, \widehat{o}_3 - 1\,]$$

Then:

$$\frac{\partial L}{\partial \mathbf{W}^{(o)}} = \begin{bmatrix} (\widehat{o}_1) \times 1 & (\widehat{o}_1) \times 2 \\ (\widehat{o}_2) \times 1 & (\widehat{o}_2) \times 2 \\ (\widehat{o}_3 - 1) \times 1 & (\widehat{o}_3 - 1) \times 2 \end{bmatrix} \tag{27}$$

2.4. **Step 3: Through the ReLU and $\mathbf{W}^{(2)}$ Layer.** We have already computed the gradients with respect to the second hidden layer's activations and the output weights:

$$\frac{\partial L}{\partial h_1^{(2)}}, \quad \frac{\partial L}{\partial h_2^{(2)}}, \quad \text{and} \quad \frac{\partial L}{\partial \mathbf{W}^{(o)}}$$

Now, in Step 3, we will move one layer backward to compute the following:

$$\frac{\partial L}{\partial h_1^{(1)}}, \quad \frac{\partial L}{\partial h_2^{(1)}}, \quad \frac{\partial L}{\partial \mathbf{W}^{(2)}}$$

This step passes the gradient signal through the **ReLU** activation function and the second layer's weights $\mathbf{W}^{(2)}$.

Step 3A: Compute $\dfrac{\partial L}{\partial \vec{\mathbf{h}}^{(1)}}$.

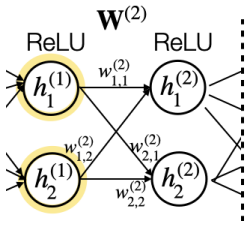

FIGURE 6. Partial network diagram highlighting hidden units in layer 1.

We now find how the loss changes with respect to the activations in the first hidden layer. From the network definition (Equation (1)), we have the following:

$$h_i^{(2)} = \mathrm{ReLU}(z_i^{(2)}), \quad \text{where} \quad z_i^{(2)} = \sum_j w_{i,j}^{(2)} h_j^{(1)} \tag{28}$$

Expanding this explicitly for our example network, we have the following:

$$\begin{aligned} h_1^{(2)} &= \mathrm{ReLU}(z_1^{(2)}) & z_1^{(2)} &= w_{1,1}^{(2)} h_1^{(1)} + w_{1,2}^{(2)} h_2^{(1)} \\ h_2^{(2)} &= \mathrm{ReLU}(z_2^{(2)}) & z_2^{(2)} &= w_{2,1}^{(2)} h_1^{(1)} + w_{2,2}^{(2)} h_2^{(1)} \end{aligned} \tag{29}$$

Let's focus on first computing $\frac{\partial L}{\partial h_1^{(1)}}$.

First, we apply the chain rule (following the pattern of Equation (4)) to get the following:

$$\frac{\partial L}{\partial h_1^{(1)}} = \frac{\partial L}{\partial \vec{\mathbf{h}}^{(2)}} \frac{\partial \vec{\mathbf{h}}^{(2)}}{h_1^{(1)}} = \begin{bmatrix} \frac{\partial L}{\partial h_1^{(2)}} & \frac{\partial L}{\partial h_2^{(2)}} \end{bmatrix} \begin{bmatrix} \frac{\partial h_1^{(2)}}{\partial h_1^{(1)}} \\ \frac{\partial h_2^{(2)}}{\partial h_1^{(1)}} \end{bmatrix}$$

$$\frac{\partial L}{\partial h_1^{(1)}} = \frac{\partial L}{\partial h_1^{(2)}} \frac{\partial h_1^{(2)}}{\partial h_1^{(1)}} + \frac{\partial L}{\partial h_2^{(2)}} \frac{\partial h_2^{(2)}}{\partial h_1^{(1)}} \tag{30}$$

We have $\frac{\partial L}{\partial \vec{\mathbf{h}}}^{(2)}$ from Step 2A (Equations (17) to (19)). Now, let's find the $\frac{\partial h_i^{(2)}}{\partial h_1^{(1)}}$terms. Using Equation (29) and the chain rule, we have the following (i.e. each $h_i^{(2)}$ depends on $h_1^{(1)}$ through $z_i^{(2)}$):

$$\frac{\partial h_i^{(2)}}{\partial h_1^{(1)}} = \frac{d\,\text{ReLU}(z_i^{(2)})}{dz_i^{(2)}} \frac{\partial z_i^{(2)}}{\partial h_1^{(1)}} \tag{31}$$

We can define the derivative of the ReLU function as follows:

$$\frac{d}{dz}\text{ReLU}(z) = \mathbb{1}[z > 0] = \begin{cases} 1, & \text{if } z > 0 \\ 0, & \text{otherwise} \end{cases} \tag{32}$$

Now, using Equation (29), we have the following:

$$\frac{\partial z_1^{(2)}}{\partial h_1^{(1)}} = w_{1,1}^{(2)} \qquad \frac{\partial z_2^{(2)}}{\partial h_1^{(1)}} = w_{2,1}^{(2)} \tag{33}$$

This can be generalized as follows:

$$\frac{\partial z_i^{(2)}}{\partial h_1^{(1)}} = w_{i,1}^{(2)} \tag{34}$$

Then, we can plug in Equations (32) and (33) into Equation (35):

$$\frac{\partial h_i^{(2)}}{\partial h_1^{(1)}} = \mathbb{1}[z_i^{(2)} > 0] \cdot w_{i,1}^{(2)} = \begin{cases} w_{i,1}^{(2)}, & \text{if } z_i^{(2)} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{35}$$

Finally, plugging in the above into Equation (36), we have what we're looking for:

$$\frac{\partial L}{\partial h_1^{(1)}} = \frac{\partial L}{\partial h_1^{(2)}} \cdot \mathbb{1}[z_1^{(2)} > 0] \cdot w_{1,1}^{(2)} + \frac{\partial L}{\partial h_2^{(2)}} \cdot \mathbb{1}[z_2^{(2)}] \cdot w_{2,1}^{(2)} \tag{36}$$

Recall that we already found $\frac{\partial L}{\partial \vec{\mathbf{h}}}^{(2)}$ from Step 2A (Equations (17) to (19)).

We can also write a similar equation for $\frac{\partial L}{\partial h_2^{(1)}}$:

$$\frac{\partial L}{\partial h_1^{(2)}} = \frac{\partial L}{\partial h_1^{(2)}} \cdot \mathbb{1}[z_1^{(2)} > 0] \cdot w_{1,2}^{(2)} + \frac{\partial L}{\partial h_2^{(2)}} \cdot \mathbb{1}[z_2^{(2)} > 0] \cdot w_{2,2}^{(2)} \tag{37}$$

Generalization. Thus, we can generalize Equations (35) and (36) as follows:

$$\frac{\partial h_i^{(2)}}{\partial h_j^{(1)}} = \mathbb{1}[z_i^{(2)} > 0] \cdot w_{i,j}^{(2)} \tag{38}$$

$$\frac{\partial L}{\partial h_j^{(1)}} = \sum_i \frac{\partial L}{\partial h_i^{(2)}} \cdot \mathbb{1}[z_i^{(2)} > 0] \cdot w_{i,j}^{(2)} \tag{39}$$

Interpretation. The ReLU activation acts as a **binary gate** for gradient flow (via $\mathbb{1}[z_i^{(2)} > 0]$): only hidden nuits with positive pre-activation ($z_i^{(2)} > 0$) pass gradients backward; inactive units block them (i.e. set gradient to 0).

Example.
Given:

$$\mathbf{W}^{(2)} = \begin{bmatrix} 1 & 3 \\ 2 & -1 \end{bmatrix}, \qquad \vec{\mathbf{h}}^{(1)} = [1, 1], \qquad \vec{\mathbf{z}}^{(2)} = [4, 1].$$

Since both pre-activations are positive:

$$z_1^{(2)} = 4 > 0, \qquad z_2^{(2)} = 1 > 0,$$

the ReLU gates are "on" for both units.
Compute $\frac{\partial h_1^{(2)}}{\partial h_2^{(1)}}$. From the layer definition:

$$h_1^{(2)} = \text{ReLU}(z_1^{(2)}), \qquad z_1^{(2)} = w_{1,1}^{(2)} h_1^{(1)} + w_{1,2}^{(2)} h_2^{(1)}.$$

Apply the chain rule:

$$\frac{\partial h_1^{(2)}}{\partial h_2^{(1)}} = \frac{d\,\text{ReLU}(z_1^{(2)})}{dz_1^{(2)}} \cdot \frac{\partial z_1^{(2)}}{\partial h_2^{(1)}}.$$

Compute each part:

$$\frac{d\,\text{ReLU}(z_1^{(2)})}{dz_1^{(2)}} = \begin{cases} 1, & z_1^{(2)} > 0, \\ 0, & \text{otherwise} \end{cases} \qquad \frac{\partial z_1^{(2)}}{\partial h_2^{(1)}} = w_{1,2}^{(2)}.$$

Combine:

$$\frac{\partial h_1^{(2)}}{\partial h_2^{(1)}} = \begin{cases} w_{1,2}^{(2)}, & z_1^{(2)} > 0, \\ 0, & \text{otherwise} \end{cases}$$

Substitute numerical values:

$$z_1^{(2)} = 4 > 0, \quad w_{1,2}^{(2)} = 3 \quad \Longrightarrow \quad \frac{\partial h_1^{(2)}}{\partial h_2^{(1)}} = 3.$$

Step 3B: Compute $\frac{\partial L}{\partial \mathbf{W}^{(2)}}$. We now compute how the loss changes with respect to the layer 2's weights $\mathbf{W}^{(2)}$.

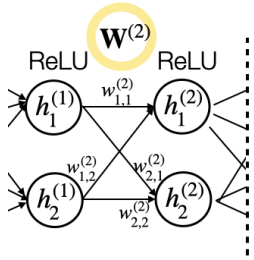

FIGURE 7. Partial network diagram highlighting layer 2's weights

Recall how we compute the activations of hidden layer 2 for our example network (copied from Equation (29)):

$$\begin{aligned} h_1^{(2)} &= \text{ReLU}(z_1^{(2)}) & z_1^{(2)} &= w_{1,1}^{(2)} h_1^{(1)} + w_{1,2}^{(2)} h_2^{(1)} \\ h_2^{(2)} &= \text{ReLU}(z_2^{(2)}) & z_2^{(2)} &= w_{2,1}^{(2)} h_1^{(1)} + w_{2,2}^{(2)} h_2^{(1)} \end{aligned} \tag{40}$$

Recall also the derivative of ReLU activation function (copied from Equation (32)):

$$\frac{d}{dz}\text{ReLU}(z) = \mathbb{1}[z > 0] = \begin{cases} 1, & \text{if } z > 0 \\ 0, & \text{otherwise} \end{cases}$$

Compute $\frac{\partial L}{\partial w_{2,1}^{(2)}}$.

Looking at Equation (40), we see that $w_{2,1}^{(2)}$ is only used in the computation of $z_2^{(2)}$ and thus $h_2^{(2)}$ (not $z_1^{(2)}$ or by extension $h_1^{(2)}$). Thus, we have the following:

$$\frac{\partial h_1^{(2)}}{\partial w_{2,1}^{(2)}} = 0 \tag{41}$$

$$\frac{\partial h_2^{(2)}}{\partial w_{2,1}^{(2)}} = \frac{d\,\text{ReLU}(z_2^{(2)})}{dz_2^{(2)}} \frac{\partial z_2^{(2)}}{\partial w_{2,1}^{(2)}} = \mathbb{1}[z_2^{(2)} > 0] \cdot h_1^{(1)} \tag{42}$$

Now, using the chain rule (following the pattern of Equation (3)) and plugging in Equations (41) and (42), we have the following:

$$\frac{\partial L}{\partial w_{2,1}^{(2)}} = \frac{\partial L}{\partial \vec{\mathbf{h}}^{(2)}} \frac{\partial \vec{\mathbf{h}}^{(2)}}{\partial w_{2,1}^{(2)}} = \begin{bmatrix} \frac{\partial L}{\partial h_1^{(2)}} & \frac{\partial L}{\partial h_2^{(2)}} \end{bmatrix} \begin{bmatrix} 0 \\ \frac{\partial h_2^{(2)}}{\partial w_{2,1}^{(2)}} \end{bmatrix} = \frac{\partial L}{\partial h_2^{(2)}} \frac{\partial h_2^{(2)}}{\partial w_{2,1}^{(2)}} \tag{43}$$

$$= \frac{\partial L}{\partial h_2^{(2)}} \cdot \mathbb{1}[z_2^{(2)} > 0] \cdot h_1^{(1)} \tag{44}$$

Recall that we already found $\frac{\partial L}{\partial \vec{\mathbf{h}}}^{(2)}$ from Step 2A (Equations (17) to (19)).

Generalization. By the same reasoning, here's all the other non-zero partial derivatives for our example network:

$$\begin{aligned} \frac{\partial h_1^{(2)}}{\partial w_{1,1}^{(2)}} &= \mathbb{1}[z_1^{(2)} > 0] \cdot h_1^{(1)} & \frac{\partial h_1^{(2)}}{\partial w_{1,2}^{(2)}} &= \mathbb{1}[z_1^{(2)} > 0] \cdot h_2^{(1)} \\ \frac{\partial h_2^{(2)}}{\partial w_{2,1}^{(2)}} &= \mathbb{1}[z_2^{(2)} > 0] \cdot h_1^{(1)} & \frac{\partial h_2^{(2)}}{\partial w_{2,2}^{(2)}} &= \mathbb{1}[z_2^{(2)} > 0] \cdot h_2^{(1)} \end{aligned} \tag{45}$$

We can generalize this for partial derivatives of the form $\frac{\partial h_i^{(2)}}{w_{i,j}^{(2)}}$:

$$\frac{\partial h_i^{(2)}}{\partial w_{i,j}^{(2)}} = \mathbb{1}[z_i^{(2)} > 0] \cdot h_j^{(1)} = \begin{cases} h_j^{(1)}, & \text{if } z_i^{(2)} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{46}$$

Thus, we have the following:

$$\frac{\partial L}{\partial w_{i,j}^{(2)}} = \frac{\partial L}{\partial h_i^{(2)}} \frac{\partial h_i^{(2)}}{\partial w_{i,j}^{(2)}} \tag{47}$$

2.5. **Step 4: Gradients w.r.t. the Input Layer Weights $\mathbf{W}^{(1)}$.** Now, we've computed all intermediate gradients up through

$$\frac{\partial L}{\partial h_1^{(1)}}, \quad \frac{\partial L}{\partial h_2^{(1)}}, \quad \frac{\partial L}{\partial \mathbf{W}^{(2)}}$$

Our final task is to compute the gradients of the loss with respect to the first layer's weights:

$$\frac{\partial L}{\partial \mathbf{W}^{(1)}}$$

These are the parameters directly connecting the input vector $\vec{\mathbf{x}} = [x_1, x_2, x_3]$ to the first hidden layer.

Network definitions. For each neuron in the first hidden layer:

$$h_i^{(1)} = \text{ReLU}(z_i^{(1)}), \quad \text{where} \quad z_i^{(1)} = \sum_j w_{i,j}^{(1)} x_j. \tag{48}$$

Expanding this explicitly for our example network, we have the following:

$$\begin{aligned} h_1^{(1)} &= \text{ReLU}(z_1^{(1)}) & z_1^{(1)} &= w_{1,1}^{(1)} x_1 + w_{1,2}^{(1)} x_2 + w_{1,3}^{(1)} x_3 \\ h_2^{(1)} &= \text{ReLU}(z_2^{(1)}) & z_2^{(1)} &= w_{2,1}^{(1)} x_1 + w_{2,2}^{(1)} x_2 + w_{2,3}^{(1)} x_3 \end{aligned} \tag{49}$$

The derivative of ReLU remains the same as before (copied from Equation (32)):

$$\frac{d}{dz}\text{ReLU}(z) = \mathbb{1}[z > 0] = \begin{cases} 1, & \text{if } z > 0 \\ 0, & \text{otherwise} \end{cases}$$

Computing $\frac{\partial L}{\partial w_{i,j}^{(1)}}$. Using the chain rule, we have the following:

$$\frac{\partial L}{\partial w_{i,j}^{(1)}} = \frac{\partial L}{\partial h_i^{(1)}} \frac{\partial h_i^{(1)}}{\partial w_{i,j}^{(1)}} \tag{50}$$

We have already found the first term (i.e. an element of $\frac{\partial L}{\partial \vec{\mathbf{h}}^{(1)}}$) in Step 3A. Let's find the second term:

$$\frac{\partial h_i^{(1)}}{\partial w_{i,j}^{(1)}} = \frac{\partial h_i^{(1)}}{\partial z_i^{(1)}} \frac{\partial z_i^{(1)}}{\partial w_{i,j}^{(1)}} = \frac{d\text{ReLU}(z_i^{(1)})}{dz_i^{(1)}} \frac{\partial z_i^{(1)}}{\partial w_{i,j}^{(1)}} \tag{51}$$

$$= \mathbb{1}[z_i^{(1)} > 0] \cdot x_j \tag{52}$$

Thus, we have the following:

$$\frac{\partial L}{\partial w_{i,j}^{(1)}} = \frac{\partial L}{\partial h_i^{(1)}} \cdot \mathbb{1}[z_i^{(1)} > 0] \cdot x_j \tag{53}$$

Notice how Step 4 mirrors the structure of Step 3B, but now the "inputs" are the raw features $\vec{\mathbf{x}}$ rather than hidden activations $\vec{\mathbf{h}}^{(1)}$. One way to think about this step is to treat $\vec{\mathbf{x}}$ as $\vec{\mathbf{h}}^{(0)}$ and then follow Step 3B but change previous (or input) layer index $l = 1$ to $l = 0$ and next layer index $l + 1 = 2$ to $l + 1 = 1$.

Summary of the Full Backward Pass. We have now computed all required gradients of the loss w.r.t. all network weights that are needed for gradient descent:

$$\frac{\partial L}{\partial \mathbf{W}^{(1)}}, \quad \frac{\partial L}{\partial \mathbf{W}^{(2)}}, \quad \frac{\partial L}{\partial \mathbf{W}^{(o)}}$$

Backpropagation proceeds step-by-step:

$$\vec{\mathbf{o}} \to \vec{\mathbf{h}}^{(2)} \to \vec{\mathbf{h}}^{(1)} \to \vec{\mathbf{x}}$$

with each layer passing its local gradient to the one before it using the chain rule. This completes one full backward pass of the neural network!

3. Backpropagation Summary

Now, we have derived all gradients step by step, from the output layer back to the input layer. Here is a summary of the key formulas for computing the relevant gradients and their interpretation, now generalized for any feed-forward neural network (i.e. not just our example 2-hidden layer network):

1. Gradients at the Output Layer. For the softmax + cross-entropy loss:

$$\frac{\partial L}{\partial o_i} = \begin{cases} -1 + \widehat{o}_i, & \text{if } i = y \quad (\text{always } < 0) \\ \widehat{o}_i, & \text{otherwise} \quad (\text{always } > 0) \end{cases} \tag{54}$$

2. Gradients for the Last Hidden Layer and Output Weights. For the last hidden layer (i.e. hidden layer 2 in our example network):

$$\frac{\partial L}{\partial h_j^{(2)}} = \sum_i \frac{\partial L}{\partial o_i} \frac{\partial o_i}{\partial h_j^{(2)}}, \quad \text{where } \frac{\partial o_i}{\partial h_j^{(2)}} = w_{i,j}^{(o)}$$
$$\Rightarrow \frac{\partial L}{\partial h_j^{(2)}} = \sum_i \frac{\partial L}{\partial o_i} w_{i,j}^{(o)} \tag{55}$$

For the output weights:

$$\frac{\partial L}{\partial w_{i,j}^{(o)}} = \frac{\partial L}{\partial o_i} \frac{\partial o_i}{\partial w_{i,j}^{(o)}}, \quad \text{where } \frac{\partial o_i}{\partial w_{i,j}^{(o)}} = \begin{cases} h_j^{(2)}, & \text{if } i = k \\ 0, & \text{otherwise} \end{cases}$$
$$\Rightarrow \frac{\partial L}{\partial w_{i,j}^{(o)}} = \frac{\partial L}{\partial o_i} h_j^{(2)} \tag{56}$$

Use these equations for the **last hidden layer** and **last weights**. Different equations apply for different non-linear activation functions (e.g. softmax vs. ReLU functions).

3. Gradients for ReLU Hidden Layers. For any hidden layer (except the last hidden layer):

$$\frac{\partial L}{\partial h_j^{(1)}} = \sum_i \frac{\partial L}{\partial h_i^{(2)}} \frac{\partial h_i^{(2)}}{\partial h_j^{(1)}}, \text{ where } \frac{\partial h_i^{(2)}}{\partial h_j^{(1)}} = \mathbb{1}[z_i^{(2)} > 0] \cdot w_{i,j}^{(2)} = \begin{cases} w_{i,j}^{(2)}, & \text{if } z_i^{(2)} > 0 \\ 0, & \text{otherwise} \end{cases}$$
$$\Rightarrow \frac{\partial L}{\partial h_j^{(1)}} = \sum_i \frac{\partial L}{\partial h_i^{(2)}} \cdot \mathbb{1}[z_i^{(2)} > 0] \cdot w_{i,j}^{(2)} \tag{57}$$

For the corresponding weights:

$$\frac{\partial L}{\partial w_{i,j}^{(2)}} = \frac{\partial L}{\partial h_i^{(2)}} \frac{\partial h_i^{(2)}}{\partial w_{i,j}^{(2)}}, \text{ where } \frac{\partial h_i^{(2)}}{\partial w_{i,j}^{(2)}} = \mathbb{1}[z_i^{(2)} > 0] \cdot h_j^{(1)} = \begin{cases} h_j^{(1)}, & \text{if } z_i^{(2)} > 0 \\ 0, & \text{otherwise} \end{cases}$$
$$\Rightarrow \frac{\partial L}{\partial w_{i,j}^{(2)}} = \frac{\partial L}{\partial h_i^{(2)}} \cdot \mathbb{1}[z_i^{(2)} > 0] \cdot h_j^{(1)} \tag{58}$$

Use these equations for **all other hidden layers** (except the last hidden layer) and their **weights**. Replace indices 1 and 2 with layer indices l and $l+1$.

4. Backpropagation Summary (Matrix View)

In this section, we show how to compute the relevant gradients using *matrices* for our example 2-hidden layer feedforward network. See Course Notes (CN) section 11.4.2 (link) for more details.

Step 1.

$$\frac{\partial L}{\partial o_i} = \begin{cases} -1 + \widehat{o}_i, & \text{if } i = y \quad (\text{always } < 0), \\ \widehat{o}_i, & \text{otherwise} \quad (\text{always } > 0). \end{cases} \tag{59}$$

$$\frac{\partial L}{\partial \vec{\mathbf{o}}} = \begin{bmatrix} \frac{\partial L}{\partial o_1} & \frac{\partial L}{\partial o_2} & \cdots & \frac{\partial L}{\partial o_K} \end{bmatrix} \qquad \text{“gradient of loss w.r.t. } \vec{\mathbf{o}} \text{ vector”} \ (\ 1 \times K \text{ matrix})$$

One-hot vector. Let $\vec{\mathbf{e}}_y$ be the one-hot vector with 1 at index y and 0 elsewhere. For example, if $K = 3$ and $y = 2$,

$$\vec{\mathbf{e}}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\frac{\partial L}{\partial \vec{\mathbf{o}}} = (\widehat{\vec{\mathbf{o}}} - \vec{\mathbf{e}}_y)^\top = \begin{bmatrix} \widehat{o}_1 & \widehat{o}_2 & \cdots & \widehat{o}_K \end{bmatrix} - \begin{bmatrix} e_{y,1} & e_{y,2} & \cdots & e_{y,K} \end{bmatrix} \qquad (1 \times K \text{ matrix}) \tag{60}$$

Step 2A.

$$\frac{\partial L}{\partial h_1^{(2)}} = \frac{\partial L}{\partial \vec{\mathbf{o}}} \frac{\partial \vec{\mathbf{o}}}{\partial h_1^{(2)}} = \begin{bmatrix} \frac{\partial L}{\partial o_1} & \frac{\partial L}{\partial o_2} & \frac{\partial L}{\partial o_3} \end{bmatrix} \begin{bmatrix} \frac{\partial o_1}{\partial h_1^{(2)}} \\ \frac{\partial o_2}{\partial h_1^{(2)}} \\ \frac{\partial o_3}{\partial h_1^{(2)}} \end{bmatrix} \quad \text{(chain rule)} \tag{61}$$

$$o_1 = w_{1,1}^{(o)} h_1^{(2)} + w_{1,2}^{(o)} h_2^{(2)} \qquad \Rightarrow \quad \frac{\partial o_1}{\partial h_1^{(2)}} = w_{1,1}^{(o)} \tag{62}$$

$$o_2 = w_{2,1}^{(o)} h_1^{(2)} + w_{2,2}^{(o)} h_2^{(2)} \qquad \Rightarrow \quad \frac{\partial o_2}{\partial h_1^{(2)}} = w_{2,1}^{(o)} \tag{63}$$

$$o_3 = w_{3,1}^{(o)} h_1^{(2)} + w_{3,2}^{(o)} h_2^{(2)} \qquad \Rightarrow \quad \frac{\partial o_3}{\partial h_1^{(2)}} = w_{3,1}^{(o)} \tag{64}$$

$$\frac{\partial \vec{\mathbf{o}}}{\partial h^{(2)}} = \begin{bmatrix} w_{1,1}^{(o)} & w_{1,2}^{(o)} \\ w_{2,1}^{(o)} & w_{2,2}^{(o)} \\ w_{3,1}^{(o)} & w_{3,2}^{(o)} \end{bmatrix} = \mathbf{W}^{(o)}$$

$$\Rightarrow \frac{\partial L}{\partial \vec{\mathbf{h}}^{(2)}} = \frac{\partial L}{\partial \vec{\mathbf{o}}} \mathbf{W}^{(o)} \tag{65}$$

Step 2B.

$$o_k = w_{k1}^{(o)} h_1^{(2)} + w_{k2}^{(o)} h_2^{(2)}, \qquad \frac{\partial o_k}{\partial w_{i,j}^{(o)}} = \begin{cases} h_j^{(2)}, & \text{if } i = k, \\ 0, & \text{otherwise} \end{cases} \tag{66}$$

$$\frac{\partial L}{\partial w_{2,1}^{(o)}} = \frac{\partial L}{\partial \vec{\mathbf{o}}} \frac{\partial \vec{\mathbf{o}}}{\partial w_{2,1}^{(o)}} = \begin{bmatrix} \frac{\partial L}{\partial o_1} & \frac{\partial L}{\partial o_2} & \frac{\partial L}{\partial o_3} \end{bmatrix} \begin{bmatrix} 0 \\ h_1^{(2)} \\ 0 \end{bmatrix} = \frac{\partial L}{\partial o_2} \times h_1^{(2)} \tag{67}$$

$$\Rightarrow \quad \frac{\partial L}{\partial w_{i,j}^{(o)}} = \frac{\partial L}{\partial o_i} \times h_j^{(2)} \tag{68}$$

$$\frac{\partial L}{\partial \mathbf{W}^{(o)}} = \begin{bmatrix} \frac{\partial L}{\partial o_1} h_1^{(2)} & \frac{\partial L}{\partial o_1} h_2^{(2)} \\ \frac{\partial L}{\partial o_2} h_1^{(2)} & \frac{\partial L}{\partial o_2} h_2^{(2)} \\ \frac{\partial L}{\partial o_3} h_1^{(2)} & \frac{\partial L}{\partial o_3} h_2^{(2)} \end{bmatrix} = \left(\frac{\partial L}{\partial \vec{\mathbf{o}}} \right)^\top (\vec{\mathbf{h}}^{(2)})^\top \tag{69}$$

Step 3A.

$$\frac{\partial L}{\partial \vec{\mathbf{h}}^{(1)}} = \frac{\partial L}{\partial \vec{\mathbf{h}}^{(2)}} \frac{\partial \vec{\mathbf{h}}^{(2)}}{\partial \vec{\mathbf{z}}^{(2)}} \frac{\partial \vec{\mathbf{z}}^{(2)}}{\partial \vec{\mathbf{h}}^{(1)}} \tag{70}$$

$$h_1^{(2)} = \mathrm{ReLU}\Big(z_1^{(2)}\Big), \quad \text{where } z_1^{(2)} = w_{1,1}^{(2)} h_1^{(1)} + w_{1,2}^{(2)} h_2^{(1)}$$
$$h_2^{(2)} = \mathrm{ReLU}\Big(z_2^{(2)}\Big), \quad \text{where } z_2^{(2)} = w_{2,1}^{(2)} h_1^{(1)} + w_{2,2}^{(2)} h_2^{(1)}$$

$$\frac{\partial h_1^{(2)}}{\partial h_1^{(1)}} = \begin{cases} w_{1,1}^{(2)}, & \text{if } z_1^{(2)} > 0, \\ 0, & \text{otherwise} \end{cases} \qquad \frac{\partial h_1^{(2)}}{\partial h_2^{(1)}} = \begin{cases} w_{1,2}^{(2)}, & \text{if } z_1^{(2)} > 0, \\ 0, & \text{otherwise} \end{cases} \tag{71}$$

$$\frac{\partial h_2^{(2)}}{\partial h_1^{(1)}} = \begin{cases} w_{2,1}^{(2)}, & \text{if } z_2^{(2)} > 0, \\ 0, & \text{otherwise} \end{cases} \qquad \frac{\partial h_2^{(2)}}{\partial h_2^{(1)}} = \begin{cases} w_{2,2}^{(2)}, & \text{if } z_2^{(2)} > 0, \\ 0, & \text{otherwise} \end{cases} \tag{72}$$

Introduce the indicator function $\mathbb{I}[\cdot]$:

$$\mathbb{I}[z > 0] = \begin{cases} 1, & \text{if } z > 0, \\ 0, & \text{otherwise} \end{cases} \qquad \Rightarrow \quad \frac{\partial \vec{\mathbf{h}}^{(2)}}{\partial \mathbf{z}^{(2)}} = \mathrm{diag}\big(\mathbb{I}[\mathbf{z}^{(2)} > 0]\big) = \begin{bmatrix} \mathbb{I}[\mathbf{z}_1^{(2)} > 0] & 0 \\ 0 & \mathbb{I}[\mathbf{z}_2^{(2)} > 0] \end{bmatrix} \tag{73}$$

$$\frac{\partial \vec{\mathbf{z}}^{(2)}}{\partial \vec{\mathbf{h}}^{(1)}} = \mathbf{W}^{(2)} = \begin{bmatrix} w_{1,1}^{(2)} & w_{1,2}^{(2)} \\ w_{2,1}^{(2)} & w_{2,2}^{(2)} \end{bmatrix} \tag{74}$$

$$\Rightarrow \quad \frac{\partial L}{\partial \vec{\mathbf{h}}^{(1)}} = \frac{\partial L}{\partial \vec{\mathbf{h}}^{(2)}} \, \mathrm{diag}\big(\mathbb{I}[\mathbf{z}^{(2)} > 0]\big) \, \mathbf{W}^{(2)} \tag{75}$$

Step 3B.

$$\frac{\partial L}{\partial w_{2,1}^{(2)}} = \frac{\partial L}{\partial \vec{\mathbf{h}}^{(2)}} \frac{\partial \vec{\mathbf{h}}^{(2)}}{\partial w_{2,1}^{(2)}} = \begin{bmatrix} \frac{\partial L}{\partial h_1^{(2)}} & \frac{\partial L}{\partial h_2^{(2)}} \end{bmatrix} \begin{bmatrix} 0 \\ \mathbb{I}[z_2^{(2)} > 0] \cdot h_1^{(1)} \end{bmatrix} = \frac{\partial L}{\partial h_2^{(2)}} \cdot \mathbb{I}[z_2^{(2)} > 0] \cdot h_1^{(1)} \tag{76}$$

$$\Rightarrow \qquad \frac{\partial L}{\partial w_{i,j}^{(2)}} = \frac{\partial L}{\partial h_i^{(2)}} \cdot \mathbb{I}[z_i^{(2)} > 0] \cdot h_j^{(1)} \tag{77}$$

$$\frac{\partial \vec{\mathbf{h}}^{(2)}}{\partial \vec{\mathbf{z}}^{(2)}} = \mathrm{diag}\big(\mathbb{I}[\mathbf{z}^{(2)} > 0]\big) = \begin{bmatrix} \mathbb{I}[\mathbf{z}_1^{(2)} > 0] & 0 \\ 0 & \mathbb{I}[\mathbf{z}_2^{(2)} > 0] \end{bmatrix}, \qquad (\vec{\mathbf{h}}^{(1)})^\top = \begin{bmatrix} h_1^{(1)} & h_2^{(1)} \end{bmatrix} \tag{78}$$

$$\frac{\partial L}{\partial \mathbf{W}^{(2)}} = \begin{bmatrix} \frac{\partial L}{\partial w_{1,1}^{(2)}} & \frac{\partial L}{\partial w_{1,2}^{(2)}} \\ \frac{\partial L}{\partial w_{2,1}^{(2)}} & \frac{\partial L}{\partial w_{2,2}^{(2)}} \end{bmatrix} = \left(\frac{\partial \vec{\mathbf{h}}^{(2)}}{\partial \vec{\mathbf{z}}^{(2)}}\right)^{\top} \left(\frac{\partial L}{\partial \vec{\mathbf{h}}^{(2)}}\right)^{\top} (\vec{\mathbf{h}}^{(1)})^{\top} = \text{diag}\big(\mathbb{I}[\mathbf{z}^{(2)} > 0]\big) \left(\frac{\partial L}{\partial \vec{\mathbf{h}}^{(2)}}\right)^{\top} (\vec{\mathbf{h}}^{(1)})^{\top} \tag{79}$$