Neural Network Architecture

Borrowing the idea from ResNet[1, 2], I use two mechanisms here, the skip connection and the pre-activation, to construct my own neural network (See Figure 1). It is supposed that we are training a multi-class classifier with the cross-entropy loss. It can be seen from Figure 1 that all the intermediate layers, except activation functions, are fully connected layers.

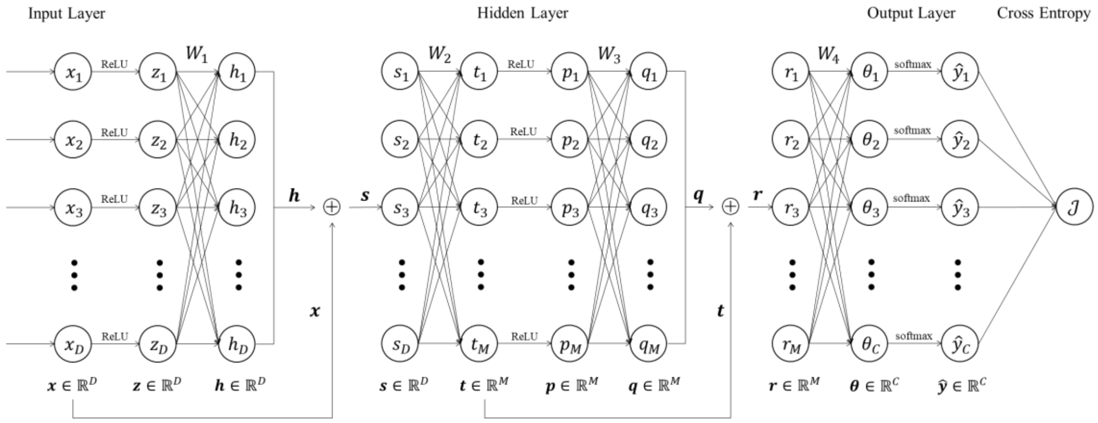

Figure 1: The architecture of this self-designed neural network.

A forward pass of a single input sample $\boldsymbol{x}$ is as follows:

$$
\begin{aligned}
&\boldsymbol{x} \in \mathbb{R}^D \\
&\boldsymbol{z} = ReLU(\boldsymbol{x}) \in \mathbb{R}^D \\
&\boldsymbol{h} = W_1 \boldsymbol{z} \in \mathbb{R}^D \\
&\boldsymbol{s} = \boldsymbol{h} + \boldsymbol{x} \in \mathbb{R}^D \\
&\boldsymbol{t} = W_2 \boldsymbol{s} \in \mathbb{R}^M \\
&\boldsymbol{p} = ReLU(\boldsymbol{t}) \in \mathbb{R}^M \\
&\boldsymbol{q} = W_3 \boldsymbol{p} \in \mathbb{R}^M \\
&\boldsymbol{r} = \boldsymbol{q} + \boldsymbol{t} \in \mathbb{R}^M \\
&\boldsymbol{\theta} = W_4 \boldsymbol{r} \in \mathbb{R}^C \\
&\hat{\boldsymbol{y}} = softmax(\boldsymbol{\theta}) \in \mathbb{R}^C \\
&\mathcal{J} = CrossEntropy(\boldsymbol{y}, \hat{\boldsymbol{y}}) \in \mathbb{R}
\end{aligned}
$$

where $W_1 \in \mathbb{R}^{D\times D}$, $W_2 \in \mathbb{R}^{M\times D}$, $W_3 \in \mathbb{R}^{M\times M}$, $W_4 \in \mathbb{R}^{C\times M}$ are weight matrices. $\boldsymbol{y} \in \mathbb{R}^C$ is the true label vector associated with $\boldsymbol{x}$, that is, a one-hot vector indicting which class $\boldsymbol{x}$ belongs to. The softmax activation of the j-th output unit is

$$\hat{y}_j = softmax(\theta_j) = \frac{e^{\theta_j}}{\sum_{j=1}^{C} e^{\theta_j}}.$$

The cross-entropy loss for this single input sample $\boldsymbol{x}$ is

$$\mathcal{J} = -\sum_{j=1}^{C} y_j log \hat{y}_j = -\boldsymbol{y}^T log \hat{\boldsymbol{y}}.$$

Gradients Derivation

First, let us start with calculating the common parts shared by all four target gradients. we can see that

$$\frac{\partial \mathcal{J}}{\partial \hat{\boldsymbol{y}}} = \frac{\partial(-\boldsymbol{y}^T log \hat{\boldsymbol{y}})}{\partial \hat{\boldsymbol{y}}} = -\left[\frac{y_1}{\hat{y}_1}, \cdots, \frac{y_C}{\hat{y}_C}\right] = -\left(\frac{\boldsymbol{y}}{\hat{\boldsymbol{y}}}\right)^T \in \mathbb{R}^{1\times C}$$

where the division in $\frac{\boldsymbol{y}}{\hat{\boldsymbol{y}}}$ is an element-wise operator.

For the gradient of softmax function, we look at a single entry first:

$$\frac{\partial \hat{y}_i}{\partial \theta_j}=\frac{\partial}{\partial \theta_j}\frac{e^{\theta_i}}{\sum_{k=1}^{C}e^{\theta_k}}=\begin{cases}\frac{e^{\theta_j}\left(\sum_{k=1}^{C}e^{\theta_k}-e^{\theta_j}\right)}{(\sum_{k=1}^{C}e^{\theta_k})^2}=\hat{y}_j\left(1-\hat{y}_j\right), i=j\\ -\frac{e^{\theta_i}e^{\theta_j}}{(\sum_{k=1}^{C}e^{\theta_k})^2}=-\hat{y}_i\hat{y}_j, i\neq j\end{cases}.$$

By chain rule, we can compute

$$\begin{aligned}\frac{\partial \mathcal{J}}{\partial \theta_j}&=\sum_{i=1}^{C}\frac{\partial \mathcal{J}}{\partial \hat{y}_i}\frac{\partial \hat{y}_i}{\partial \theta_j}=-\sum_{i=1}^{C}\frac{y_i}{\hat{y}_i}\frac{\partial \hat{y}_i}{\partial \theta_j}=-\left(\frac{y_j}{\hat{y}_j}\frac{\partial \hat{y}_j}{\partial \theta_j}+\sum_{i\neq j}\frac{y_i}{\hat{y}_i}\frac{\partial \hat{y}_i}{\partial \theta_j}\right)\\&=-\frac{y_j}{\hat{y}_j}\hat{y}_j\left(1-\hat{y}_j\right)+\sum_{i\neq j}\frac{y_i}{\hat{y}_i}\hat{y}_i\hat{y}_j=-y_j\left(1-\hat{y}_j\right)+\hat{y}_j\sum_{i\neq j}y_i\\&=-y_j+\hat{y}_j\sum_{i=1}^{C}y_i.\end{aligned}$$

Recall that $\boldsymbol{y}$ is a one-hot vector, that is, $\sum_{i=1}^{C}y_i=1$, we can get

$$\frac{\partial \mathcal{J}}{\partial \theta_j}=\hat{y}_j-y_j$$

and

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\theta}}=(\hat{\boldsymbol{y}}-\boldsymbol{y})^T\in\mathbb{R}^{1\times C}.$$

Now we examine the gradient of ReLU function. Recall that $ReLU(x)=\max(x,0)$, by extending that its derivative at $x=0$ equals 0, we get

$$ReLU'(x)=\begin{cases}1, x>0\\0, otherwise\end{cases}=\mathbb{I}_{x>0}(x)\in\{0,1\}$$

where $\mathbb{I}_{x>0}(x)$ is an indicator function. Furthermore, we can extend ReLU function to a vector-valued function by applying ReLU to each entry of an input vector $\boldsymbol{x}\in\mathbb{R}^D$ and calculate its gradient as

$$\frac{\partial ReLU(\boldsymbol{x})}{\partial \boldsymbol{x}}=diag\left(\mathbb{I}_{x_1>0}(x_1),\cdots,\mathbb{I}_{x_D>0}(x_D)\right)\in\mathbb{R}^{D\times D}.$$

It is easy to see that W_1, W_2, W_3 and W_4 contain all the network parameters. So we just need to derive their gradients to update the whole neural network.

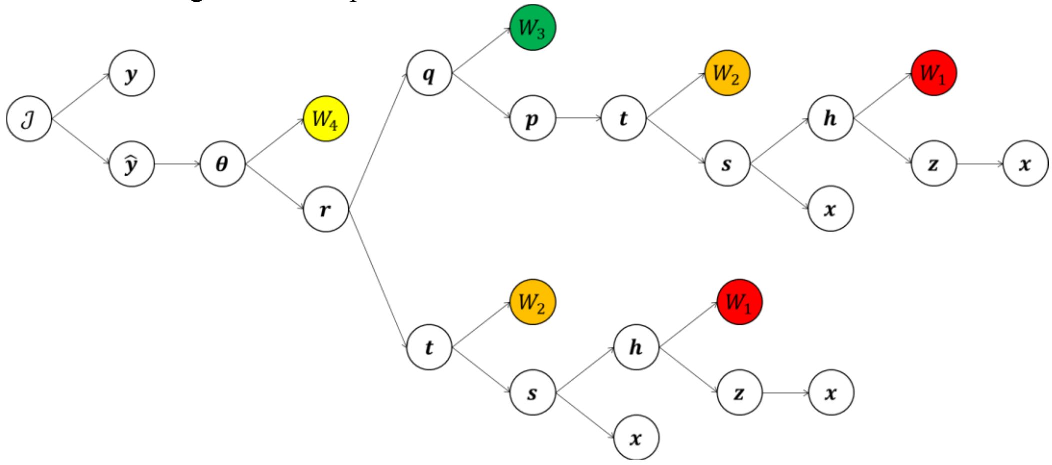

Figure 2: The computation graph of this self-designed neural network.

However, due to the complexity of this neural network, it is hard to capture directly how the gradients flow just by observing its architecture figure. So I show the computation graph of it to make the process of gradients derivation more clearly (See Figure 2).

It can be seen obviously from Figure 2 that there is only one path to approach W_3 and W_4, but two to approach W_1 and W_2. With the help of this computation graph, we can write down the gradients for all the network parameters by using the chain rule as

$$\frac{\partial \mathcal{J}}{\partial W_4} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{\theta}} \frac{\partial \boldsymbol{\theta}}{\partial W_4}$$
$$\frac{\partial \mathcal{J}}{\partial W_3} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{r}} \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{q}} \frac{\partial \boldsymbol{q}}{\partial W_3} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{\theta}} \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{r}} \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{q}} \frac{\partial \boldsymbol{q}}{\partial W_3}$$
$$\frac{\partial \mathcal{J}}{\partial W_2} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{r}} \left(\left(\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{q}} \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{p}} \frac{\partial \boldsymbol{p}}{\partial \boldsymbol{t}} \right)_1 \frac{\partial \boldsymbol{t}}{\partial W_2} + \left(\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{t}} \right)_2 \frac{\partial \boldsymbol{t}}{\partial W_2} \right)$$
$$\frac{\partial \mathcal{J}}{\partial W_1} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{r}} \left(\left(\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{q}} \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{p}} \frac{\partial \boldsymbol{p}}{\partial \boldsymbol{t}} \right)_1 \frac{\partial \boldsymbol{t}}{\partial \boldsymbol{s}} \frac{\partial \boldsymbol{s}}{\partial \boldsymbol{h}} \frac{\partial \boldsymbol{h}}{\partial W_1} + \left(\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{t}} \right)_2 \frac{\partial \boldsymbol{t}}{\partial \boldsymbol{s}} \frac{\partial \boldsymbol{s}}{\partial \boldsymbol{h}} \frac{\partial \boldsymbol{h}}{\partial W_1} \right).$$

Subscript 1 means the gradient flows through the upper branch of the computation graph and subscript 2 the lower. We use subscripts here to avoid confusion, that is, $\left(\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{q}} \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{p}} \frac{\partial \boldsymbol{p}}{\partial \boldsymbol{t}} \right)_1 \neq \left(\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{t}} \right)_2$.

It is noticed that $\frac{\partial \mathcal{J}}{\partial \boldsymbol{r}}$, $\left(\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{q}} \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{p}} \frac{\partial \boldsymbol{p}}{\partial \boldsymbol{t}} \right)_1$, $\frac{\partial \boldsymbol{t}}{\partial \boldsymbol{s}} \frac{\partial \boldsymbol{s}}{\partial \boldsymbol{h}}$ and $\left(\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{t}} \right)_2$ are used multiple times, so it is convenient to calculate them firstly:

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{r}} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{\theta}} \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{r}} = (\hat{\boldsymbol{y}} - \boldsymbol{y})^T W_4 \in \mathbb{R}^{1\times M}$$
$$\left(\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{q}} \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{p}} \frac{\partial \boldsymbol{p}}{\partial \boldsymbol{t}} \right)_1 = I_M W_3 diag\left(\mathbb{I}_{t_1>0}(t_1), \cdots, \mathbb{I}_{t_M>0}(t_M) \right) = W_3 \Lambda_{\boldsymbol{t}} \in \mathbb{R}^{M\times M}$$
$$\frac{\partial \boldsymbol{t}}{\partial \boldsymbol{s}} \frac{\partial \boldsymbol{s}}{\partial \boldsymbol{h}} = W_2 I_D = W_2 \in \mathbb{R}^{M\times D}$$
$$\left(\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{t}} \right)_2 = I_M \in \mathbb{R}^{M\times M}$$

where $\Lambda_{\boldsymbol{t}} = diag\left(\mathbb{I}_{t_1>0}(t_1), \cdots, \mathbb{I}_{t_M>0}(t_M) \right)$.

Now we can rewrite the target gradients as

$$\frac{\partial \mathcal{J}}{\partial W_4} = (\hat{\boldsymbol{y}} - \boldsymbol{y})^T \frac{\partial \boldsymbol{\theta}}{\partial W_4}$$
$$\frac{\partial \mathcal{J}}{\partial W_3} = (\hat{\boldsymbol{y}} - \boldsymbol{y})^T W_4 \frac{\partial \boldsymbol{q}}{\partial W_3}$$
$$\frac{\partial \mathcal{J}}{\partial W_2} = (\hat{\boldsymbol{y}} - \boldsymbol{y})^T W_4 W_3 \Lambda_{\boldsymbol{t}} \frac{\partial \boldsymbol{t}}{\partial W_2} + (\hat{\boldsymbol{y}} - \boldsymbol{y})^T W_4 \frac{\partial \boldsymbol{t}}{\partial W_2}$$
$$\frac{\partial \mathcal{J}}{\partial W_1} = (\hat{\boldsymbol{y}} - \boldsymbol{y})^T W_4 W_3 \Lambda_{\boldsymbol{t}} W_2 \frac{\partial \boldsymbol{h}}{\partial W_1} + (\hat{\boldsymbol{y}} - \boldsymbol{y})^T W_4 W_2 \frac{\partial \boldsymbol{h}}{\partial W_1}.$$

Then we continue to calculate the rest parts. For the gradient $\frac{\partial \mathcal{J}}{\partial W_4}$, let us start by analyzing the gradient $\frac{\partial \boldsymbol{\theta}}{\partial W_4}$. By definition, this gradient is the collection of the partial derivatives:

$$\frac{\partial \boldsymbol{\theta}}{\partial W_4} = \begin{bmatrix} \frac{\partial \theta_1}{\partial W_4} \\ \vdots \\ \frac{\partial \theta_C}{\partial W_4} \end{bmatrix} \in \mathbb{R}^{C \times C \times M}, \qquad \frac{\partial \theta_i}{\partial W_4} \in \mathbb{R}^{C \times M}.$$

We can write W_4 in this form:

$$W_4 = \begin{bmatrix} \boldsymbol{w}_1^{(4)} \\ \vdots \\ \boldsymbol{w}_C^{(4)} \end{bmatrix}$$

where $\boldsymbol{w}_i^{(4)} \in \mathbb{R}^{1 \times M}$ is the i-th row of W_4. So we can explicitly write out $\frac{\partial \theta_i}{\partial W_4}$ as

$$\frac{\partial \theta_i}{\partial W_4} = \frac{\partial}{\partial W_4} \boldsymbol{w}_i^{(4)} \boldsymbol{r} = \begin{bmatrix} \boldsymbol{0}^T \\ \vdots \\ \boldsymbol{0}^T \\ \boldsymbol{r}^T \\ \boldsymbol{0}^T \\ \vdots \\ \boldsymbol{0}^T \end{bmatrix} \in \mathbb{R}^{C \times M}$$

that is, a matrix with i-th row being $\boldsymbol{r}^T$ and other rows being $\boldsymbol{0}^T$. Combining them together, we can get

$$\frac{\partial \mathcal{J}}{\partial W_4} = (\hat{\boldsymbol{y}} - \boldsymbol{y})^T \frac{\partial \boldsymbol{\theta}}{\partial W_4} = (\hat{\boldsymbol{y}} - \boldsymbol{y})^T \begin{bmatrix} \frac{\partial \theta_1}{\partial W_4} \\ \vdots \\ \frac{\partial \theta_C}{\partial W_4} \end{bmatrix} = \begin{bmatrix} (\hat{\boldsymbol{y}} - \boldsymbol{y})^T \frac{\partial \theta_1}{\partial W_4} \\ \vdots \\ (\hat{\boldsymbol{y}} - \boldsymbol{y})^T \frac{\partial \theta_C}{\partial W_4} \end{bmatrix} = \begin{bmatrix} (\hat{y}_1 - y_1)\boldsymbol{r}^T \\ \vdots \\ (\hat{y}_C - y_C)\boldsymbol{r}^T \end{bmatrix} = (\hat{\boldsymbol{y}} - \boldsymbol{y})\boldsymbol{r}^T.$$

Similarly, we can get that

$$\begin{aligned}
\frac{\partial \mathcal{J}}{\partial W_3} &= (\hat{\boldsymbol{y}} - \boldsymbol{y})^T W_4 \frac{\partial \boldsymbol{q}}{\partial W_3} = {W_4}^T (\hat{\boldsymbol{y}} - \boldsymbol{y}) \boldsymbol{p}^T \\
\frac{\partial \mathcal{J}}{\partial W_2} &= (\hat{\boldsymbol{y}} - \boldsymbol{y})^T W_4 W_3 \Lambda_{\boldsymbol{t}} \frac{\partial \boldsymbol{t}}{\partial W_2} + (\hat{\boldsymbol{y}} - \boldsymbol{y})^T W_4 \frac{\partial \boldsymbol{t}}{\partial W_2} \\
&= \Lambda_{\mathbf{t}} {W_3}^T {W_4}^T (\hat{\boldsymbol{y}} - \boldsymbol{y}) \boldsymbol{s}^T + {W_4}^T (\hat{\boldsymbol{y}} - \boldsymbol{y}) \boldsymbol{s}^T \\
\frac{\partial \mathcal{J}}{\partial W_1} &= (\hat{\boldsymbol{y}} - \boldsymbol{y})^T W_4 W_3 \Lambda_{\boldsymbol{t}} W_2 \frac{\partial \boldsymbol{h}}{\partial W_1} + (\hat{\boldsymbol{y}} - \boldsymbol{y})^T W_4 W_2 \frac{\partial \boldsymbol{h}}{\partial W_1}. \\
&= {W_2}^T \Lambda_{\mathbf{t}} {W_3}^T {W_4}^T (\hat{\boldsymbol{y}} - \boldsymbol{y}) \boldsymbol{z}^T + {W_2}^T {W_4}^T (\hat{\boldsymbol{y}} - \boldsymbol{y}) \boldsymbol{z}^T.
\end{aligned}$$

At the end, we show the final results as follows:

$$\begin{aligned}
\frac{\partial \mathcal{J}}{\partial W_4} &= (\hat{\boldsymbol{y}} - \boldsymbol{y})\boldsymbol{r}^T \\
\frac{\partial \mathcal{J}}{\partial W_3} &= {W_4}^T (\hat{\boldsymbol{y}} - \boldsymbol{y}) \boldsymbol{p}^T \\
\frac{\partial \mathcal{J}}{\partial W_2} &= (\Lambda_{\mathbf{t}} {W_3}^T + I_M) {W_4}^T (\hat{\boldsymbol{y}} - \boldsymbol{y}) \boldsymbol{s}^T \\
\frac{\partial \mathcal{J}}{\partial W_1} &= {W_2}^T (\Lambda_{\mathbf{t}} {W_3}^T + I_M) {W_4}^T (\hat{\boldsymbol{y}} - \boldsymbol{y}) \boldsymbol{z}^T.
\end{aligned}$$

Training Equations

In real practice, we never use just a single input to update the neural network. Instead, we train a neural network with a training set in most cases. To make this assignment more practical, we suppose that we use the training set $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$ to train the neural network.

As a result, the final loss function needs to be modified as

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} \mathcal{J}_i(\boldsymbol{x}_i, \boldsymbol{y}_i)$$

where $\mathcal{J}_i(\boldsymbol{x}_i, \boldsymbol{y}_i)$ is the cross-entropy loss w.r.t a single input $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, whose gradients are shown in the above section. And the gradients of it are

$$\frac{\partial \mathcal{L}}{\partial W_4} = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial \mathcal{J}_i}{\partial W_4} = \frac{1}{N}\sum_{i=1}^{N} (\widehat{\boldsymbol{y}}_i - \boldsymbol{y}_i)\boldsymbol{r}_i^T$$

$$\frac{\partial \mathcal{L}}{\partial W_3} = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial \mathcal{J}_i}{\partial W_3} = \frac{1}{N} {W_4}^T \sum_{i=1}^{N} (\widehat{\boldsymbol{y}}_i - \boldsymbol{y}_i)\boldsymbol{p}_i^T$$

$$\frac{\partial \mathcal{L}}{\partial W_2} = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial \mathcal{J}_i}{\partial W_2} = \frac{1}{N}\sum_{i=1}^{N} (\Lambda_{\mathbf{t}_i} {W_3}^T + I_M){W_4}^T(\widehat{\boldsymbol{y}}_i - \boldsymbol{y}_i)\boldsymbol{s}_i^T$$

$$\frac{\partial \mathcal{L}}{\partial W_1} = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial \mathcal{J}_i}{\partial W_1} = \frac{1}{N} {W_2}^T \sum_{i=1}^{N} (\Lambda_{\mathbf{t}_i} {W_3}^T + I_M){W_4}^T(\widehat{\boldsymbol{y}}_i - \boldsymbol{y}_i)\boldsymbol{z}_i^T.$$

For the training process, we start by randomly generating $W_1 \sim W_4$. At step t, we use the following training equations to update the parameters:

$$W_4^t \leftarrow W_4^{t-1} - \alpha \frac{\partial \mathcal{L}}{\partial W_4}$$

$$W_3^t \leftarrow W_3^{t-1} - \alpha \frac{\partial \mathcal{L}}{\partial W_3}$$

$$W_2^t \leftarrow W_2^{t-1} - \alpha \frac{\partial \mathcal{L}}{\partial W_2}$$

$$W_1^t \leftarrow W_1^{t-1} - \alpha \frac{\partial \mathcal{L}}{\partial W_1}$$

where α is the learning rate.

Reference

[1] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[2] He, Kaiming, et al. "Identity mappings in deep residual networks." European conference on computer vision. Springer, Cham, 2016.