1. **Backpropagation through time (BPTT) in RNNs**

After the RNN outputs the prediction vector $h(t)$, we compute the prediction error $E(t)$ and use the Back Propagation Through time algorithm to compute the gradient

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial W}. \tag{1}$$

The gradient is used to update the model parameters by:

$$W \leftarrow W - \alpha \frac{\partial E}{\partial W}. \tag{2}$$

And we continue the learning process using the Gradient Descent algorithm.

Say we have learning task that includes T time steps, the gradient of the error on the k time step is given by:

$$\frac{\partial E_k}{\partial W} = \frac{\partial E}{\partial h_k}\frac{\partial h_k}{\partial c_k}\cdots\frac{\partial c_2}{\partial c_1}\frac{\partial c_1}{\partial W} = \frac{\partial E_k}{\partial h_k}\frac{\partial h_k}{\partial c_k}\left(\prod_{t=2}^{k}\frac{\partial c_t}{\partial c_{t-1}}\right)\frac{\partial c_1}{\partial W} \tag{3}$$

Notice that since $W = [W_{hh}, W_{hx}]$, $c(t)$ can be written as:

$$c(t) = \tanh(W_{hh}c_{t-1} + W_{hx}x_t). \tag{4}$$

Compute the derivative of $c(t)$ and get:

$$\frac{\partial c_t}{\partial c_{t-1}} = \tanh'(W_{hh}c_{t-1} + W_{hx}x_t)\frac{\partial}{\partial c_{t-1}}[W_{hh}c_{t-1} + W_{hx}x_t] = \tanh'(W_{hh}c_{t-1} + W_{hx}x_t)W_{hh} \tag{5}$$

Plug Eq. (5) into Eq. (3) and get our backpropagated gradient

$$\frac{\partial E_k}{\partial W} = \frac{\partial E_k}{\partial h_k}\frac{\partial h_k}{\partial c_k}\left(\prod_{t=2}^{k}\tanh'(W_{hh}c_{t-1} + W_{hx}x_t)W_{hh}\right)\frac{\partial c_1}{\partial W} \tag{6}$$

The last expression tends to vanish when k is large, this is due to the derivative of the tanh activation function which is smaller than 1.
The product of derivatives can also explode if the weights W_{hh} are large enough to overpower the smaller tanh derivative, this is known as the exploding gradient problem.

2. **Long Short-Term Memory**

An LSTM network has an input vector $[h(t-1), x(t)]$ at time step t. The network cell state is denoted by $c(t)$. The output vectors passed through the network between consecutive time steps t, $t+1$ are denoted by $h(t)$.
An LSTM network has three gates that update and control the cell states, these are the forget gate, input gate and output gate. The gates use hyperbolic tangent and sigmoid activation functions. The forget gate controls what information in the cell state to forget, given new information than entered the network.

The forget gate's output is given by:

$$f_t = \sigma(W_{fh}h_{t-1} + W_{fx}x_t). \tag{7}$$

The input gate controls what new information will be encoded into the cell state, given the new input information.

The input gate's output has the form:

$$\tanh(W_{hh}h_{t-1} + W_{hx}x_t) \otimes \sigma(W_{ih}h_{t-1} + W_{ix}x_t), \tag{8}$$

where we can define

$$\tilde{c}_t = \tanh(W_{hh}h_{t-1} + W_{hx}x_t) \tag{9}$$

and

$$i_t = \sigma(W_{ih}h_{t-1} + W_{ix}x_t). \tag{10}$$

The output gate controls what information encoded in the cell state is sent to the network as input in the following time step, this is done via the output vector $h(t)$. The output gate's activations are given by:

$$o_t = \sigma(W_{oh}h_{t-1} + W_{ox}x_t) \tag{11}$$

and the cell's output vector is given by:

$$h_t = o_t \otimes tanh(c_t) \tag{12}$$

The long term dependencies and relations are encoded in the cell state vectors and it's the cell state derivative that can prevent the LSTM gradients from vanishing. The LSTM cell state has the form:

$$c_t = c_{t-1} \otimes f_t + \tilde{c}_t \otimes i_t. \tag{13}$$

2.1. **Backpropagation through time in LSTMs.** As in the RNN model, our LSTM network outputs a prediction vector $h(k)$ on the k-th time step. The knowledge encoded in the state vectors $c(t)$ captures long-term dependencies and relations in the sequential data.

The length of the data sequences can be hundreds and even thousands of time steps, making it extremely difficult to learn using a basic RNN. We compute the gradient used to update the network parameters, the computation is done over T time steps.

As in RNNs, the error term gradient is given by the following sum of T gradients:

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial W}. \tag{14}$$

The gradient of the error for some time step k has the form:

$$\frac{\partial E_k}{\partial W} = \frac{\partial E}{\partial h_k}\frac{\partial h_k}{\partial c_k} \cdots \frac{\partial c_2}{\partial c_1}\frac{\partial c_1}{\partial W} = \frac{\partial E_k}{\partial h_k}\frac{\partial h_k}{\partial c_k}\left(\prod_{t=2}^{k} \frac{\partial c_t}{\partial c_{t-1}}\right)\frac{\partial c_1}{\partial W} \tag{15}$$

As we have seen, the following product causes the gradients to vanish:

$$\prod_{t=2}^{k} \frac{\partial c_t}{\partial c_{t-1}} \tag{16}$$

In an LTSM, the state vector $c(t)$, has the form:

$$c_t = c_{t-1} \otimes f_t + \tilde{c}_t \otimes i_t. \tag{17}$$

Notice that the state vector $c(t)$ is a function of the following elements, which should be taken into account when computing the derivative during backpropagation:

$$c_{t-1}, \quad f_t, \quad \tilde{c}_t, \quad i_t \tag{18}$$

Compute the derivative of Eq. (17) and get:

$$\frac{\partial c_t}{\partial c_{t-1}} = \frac{\partial}{\partial c_{t-1}}[c_{t-1} \otimes f_t + \tilde{c}_t \otimes i_t] = \frac{\partial}{\partial c_{t-1}}[c_{t-1} \otimes f_t] + \frac{\partial}{\partial c_{t-1}}[\tilde{c}_t \otimes i_t] \tag{19}$$

$$= \frac{\partial f_t}{\partial c_{t-1}} c_{t-1} + f_t + \frac{\partial i_t}{\partial c_{t-1}} \tilde{c}_t + \frac{\partial \tilde{c}_t}{\partial c_{t-1}} i_t, \tag{20}$$

where the four derivative terms are respectively denoted as A_t, B_t, C_t and D_t. The LSTM states gradient can be written as

$$\frac{\partial E_k}{\partial W} = \frac{\partial E_k}{\partial h_k} \frac{\partial h_k}{\partial c_k} \left(\prod_{t=2}^{k} [A_t + B_t + C_t + D_t] \right) \frac{\partial c_1}{\partial W} \tag{21}$$

Notice that the gradient contains the forget gate's vector of activations, which allows the network to better control the gradients values, at each time step, using suitable parameter updates of the forget gate. The presence of the forget gate's activations allows the LSTM to decide, at each time step, that certain information should not be forgotten and to update the model's parameters accordingly.

Say that for some time step $k < T$, we have that: $\sum_{i=1}^{k} \frac{\partial E_t}{\partial W} \to 0$. Then for the gradient not to vanish, we can find a suitable parameter update of the forget gate at time step $k+1$ such that $\frac{\partial E_{k+1}}{\partial W} \not\to 0$.

It is the presence of the forget gate's vector of activations in the gradient term along with additive structure which allows the LSTM to find such a parameter update at any time step, and this yields: $\sum_{i=1}^{k+1} \frac{\partial E_t}{\partial W} \not\to 0$, and the gradient doesn't vanish.

Another important property to notice is that the cell state gradient is an additive function made up from four elements denoted $A(t)$, $B(t)$, $C(t)$, $D(t)$. This additive property enables better balancing of gradient values during backpropagation. The LSTM updates and balances the values of the four components making it more likely the additive expression does not vanish. This additive property is different from the RNN case where the gradient contained a single element inside the product.

In LSTMs, however, the presence of the forget gate, along with the additive property of the cell state gradients, enables the network to update the parameter in such a way that the different sub gradients do not necessarily agree and behave in a similar manner, making it less likely that all of the T gradients will vanish, or in other words, the series of functions does not converge to zero, and our gradients do not vanish.

Summing up, we have seen that RNNs suffer from vanishing gradients and caused by long series of multiplications of small values, diminishing the gradients and causing the learning process to become degenerate. In a analogues way, RNNs suffer from exploding gradients affected from large gradient values and hampering the learning process.

LSTMs solve the problem using a unique additive gradient structure that includes direct access to the forget gate's activations, enabling the network to encourage desired behaviour from the error gradient using frequent gates update on every time step of the learning process.