Q. Evaluate RNN, LSTM and GRU performance with the help of equations and comment on their training time.

1. Recurrent Neural Network (RNN)

Equation:

For a standard RNN, the forward pass is as follows:

$t = 1$ to $t = \tau$, we apply the following update equations:

$$
\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}) \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} \\
\hat{\boldsymbol{y}}^{(t)} &= \operatorname{softmax}(\boldsymbol{o}^{(t)})
\end{aligned}
$$

Where:

- Ht is the hidden state at time step t,
- $t-1$ is the hidden state at the previous time step,
- xt is the input at time step t,
- b and c are bias terms for the hidden and output layers,
- yt is the output at time step t.

Training Time:

RNNs are relatively fast to train compared to LSTMs and GRUs due to their simpler structure. However, their performance deteriorates on long sequences due to vanishing gradients, requiring more epochs to capture long-term dependencies.

2. Long Short-Term Memory (LSTM)

Equation:

LSTM networks are an improvement over RNNs and have the following key gates: **forget gate**, **input gate**, and **output gate**.

The LSTM equations are:

- **Forget Gate:**

$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f)$$

$$w_f = Weight$$
$$h_{t-1} = Output\ from\ previous\ timestamp$$
$$x_t = New\ input$$
$$b_f = Bias$$

- **Input Gate:**

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i)$$

- **Candidate Memory Cell:**

$$c\tilde{}_t = tanh(w_c[h_{t-1}, x_t] + b_c)$$

- **Output Gate:**

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o)$$

- **Cell State Update:**

$$c_t = f_t * c_{t-1} + i_t{}^* \, c\tilde{}_t$$

- **Hidden State Update:**

$$h_t = o_t{}^*\mathrm{tanh}(c_t)$$

The cell state Ct is a key feature, allowing LSTMs to retain information over long sequences, effectively mitigating the vanishing gradient problem.

Training Time:

LSTMs are computationally heavier due to additional gates and memory cells, resulting in longer training times compared to standard RNNs. Despite this, they excel in handling long-term dependencies, making them ideal for applications like language modeling and speech recognition.

3. Gated Recurrent Unit (GRU)

Equation:

The GRU simplifies the LSTM by combining the forget and input gates into a single **update gate**. The equations are as follows:

- Update Gate:

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

- Reset Gate:

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

- Candidate Hidden State:

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

- Hidden State Update:

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

GRU's update mechanism combines the functions of the input and forget gates in an LSTM, resulting in a more compact and computationally efficient architecture.

Training Time:

GRUs, with their simpler structure (no separate cell state), train faster than LSTMs. GRUs are efficient for moderate-length sequences and are preferred when computational resources are limited or when faster training is prioritized. They can handle some long-term dependencies but may not capture them as effectively as LSTMs in certain complex applications.

Performance Evaluation:

- **Memory**: GRUs combine the forget and input gates, leading to fewer parameters than LSTMs. Like LSTMs, GRUs help in capturing long-range dependencies by updating the hidden state using both the current input and past hidden state.

- **Training Time**: GRUs are computationally more efficient than LSTMs because they have fewer gates and parameters. As a result, they typically require less memory and training time.
- **Limitations**: Although simpler and faster than LSTMs, GRUs may not always perform as well in all tasks. However, in many cases, they offer a good trade-off between performance and computational efficiency.