Quantum Long short term memory

Shivalee RK Shah
B20ME071
`shah.14@iitj.ac.in`

Abstract

This report presents a comparative analysis between LSTM (Long Short-Term Memory), a popular recurrent neural network model, and Quantum Classical LSTM (QLSTM) models, which leverages the principles of quantum computing to enhance performance, for sentiment analysis on the IMDB dataset and part-of-speech (POS) tagging tasks. The objective was to observe the impact of QLSTM on training time and accuracies in these natural language processing tasks. The IMDB dataset, containing movie reviews with sentiment labels, was used for sentiment analysis, while a self-made POS tagging dataset was employed for the POS tagging task. The experiments involved training both models on the respective datasets and evaluating their performance in terms of training time and accuracy metrics. Results indicated that the training times were observed to be longer for QLSTM due to the computational overhead of quantum computations. In POS tagging, both models demonstrated comparable accuracies, with LSTM exhibiting faster training times. These findings provide insights into the potential benefits and trade-offs associated with QLSTM in comparison to traditional LSTM models for sentiment analysis and POS tagging tasks.

1 Model Architecture

The Quantum LSTM model is as follows:

1. Initialization: The QLSTM model is initialized with parameters such as input size, hidden size, number of qubits, number of qubit layers, backend for quantum computations, and other configuration options.

2. Quantum Device and Wires: Quantum devices and wires are defined for each component of the LSTM cell (forget gate, input gate, update gate, and output gate). The device specifies the backend for executing quantum computations, such as "default.qubit", "qiskit.basicaer", or "qiskit.ibm". The wires represent the qubits used for each component.

3. Quantum Circuit Definitions: Quantum circuits are defined for each component of the LSTM cell (forget gate, input gate, update gate, and output gate). Each quantum circuit incorporates angle embedding and basic entangler layers. These circuits are used to process the inputs and weights and produce output values for each component.

4. Quantum Node Definitions: QNodes are defined for each quantum circuit using the defined quantum device and interface "torch". QNodes provide a way to execute quantum circuits and obtain measurement results as differentiable tensors in PyTorch.

5. Weight Shapes: The shapes of the weights for each QNode are defined based on the number of qubit layers and qubits.

6. Linear Layers: Linear layers are defined to transform the concatenated input and hidden state to match the qubit dimensions and to map the qubit outputs to the hidden size.

7. Forward Pass: The forward method takes input sequences (x) and optional initial states (init_states) as input. The input sequences have shape (batch_size, seq_length, feature_size). The hidden state (h_t) and cell state (c_t) are initialized or provided as input. For each time step (t) in the sequence, the following steps are performed:

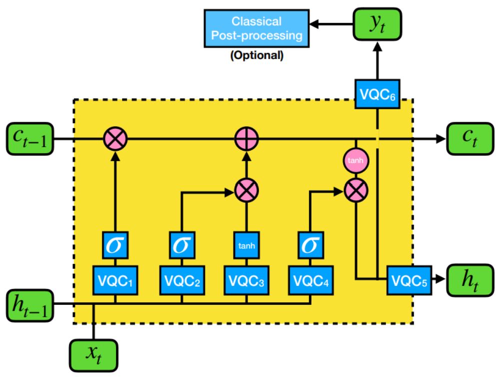


Figure 1: QLSTM model architecture

(a) Get the features for the current time step. Concatenate the input and hidden state.

(b) Apply linear transformation to match the qubit dimensions. Pass the transformed input through the QNodes corresponding to the forget gate, input gate, update gate, and output gate.

(c) Apply activation functions (sigmoid and tanh) to the QNode outputs to obtain the values for the forget gate (f_t), input gate (i_t), update gate (g_t), and output gate (o_t).

(d) Update the cell state (c_t) and hidden state (h_t) using the computed gate values and the current cell and hidden states.

(e) Append the hidden state to the list of hidden states for each time step.

8. Output: The hidden states are concatenated, transposed, and returned as the output sequence. The final hidden state and cell state are also returned as a tuple.

1.1 Comparison with LSTM

1. Representation of Information: In classical models, information is represented using classical bits, which can take on values of 0 or 1. On the other hand, quantum models use quantum bits or qubits, which can exist in a superposition of 0 and 1, enabling them to represent and process information in a more complex and probabilistic manner.

2. Computing Paradigm: Classical models follow a sequential and deterministic computing paradigm, where computations are performed sequentially and the outcome is deterministic. Quantum models, on the other hand, leverage quantum phenomena such as superposition, entanglement, and interference to perform computations in parallel on multiple qubits, potentially providing exponential speedup for certain types of problems.

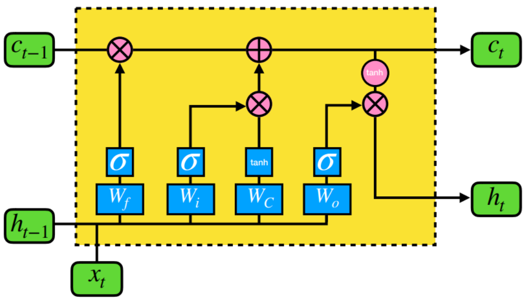


Figure 2: LSTM model architecture

3. Information Processing: Classical models process information using classical logic gates, which operate on classical bits through operations such as AND, OR, and NOT. In contrast, quantum models use quantum gates, which manipulate the state of qubits through operations such as quantum superpositions, rotations, and entangling operations.

$$
\begin{aligned}
f_t &= \sigma(V_{\text{QC1}}(v_t)) \\
i_t &= \sigma(V_{\text{QC2}}(v_t)) \\
\tilde{C}_t &= \tanh(V_{\text{QC3}}(v_t)) \\
c_t &= f_t \cdot c_{t-1} + i_t \cdot \tilde{C}_t \\
o_t &= \sigma(V_{\text{QC4}}(v_t)) \\
h_t &= V_{\text{QC5}}(o_t \cdot \tanh(c_t)) \\
y_t &= V_{\text{QC6}}(o_t \cdot \tanh(c_t))
\end{aligned}
$$

4. Quantum Effects: Quantum models can leverage unique quantum effects, such as superposition and entanglement, to perform computations that are not efficiently achievable by classical models. These effects allow for parallel processing of information, exploring multiple possibilities simultaneously, and potentially solving certain problems more efficiently than classical counterparts.

5. Computational Power: Quantum models have the potential to provide exponential

speedup for certain computational tasks compared to classical models.

2 Results and Observations

2.1 POS Tagging

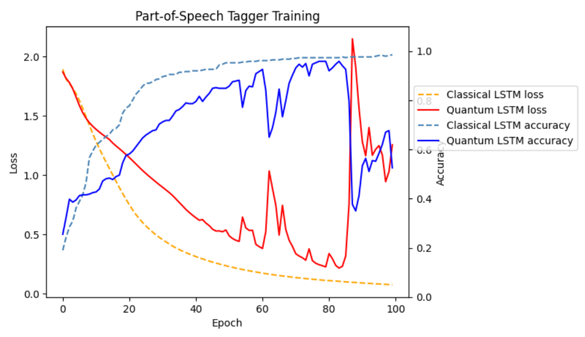
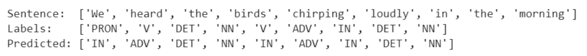
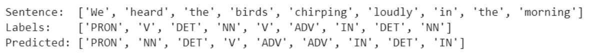


Figure 3: Loss and Accuracy curves for LSTM and QLSTM



```
Sentence:  ['We', 'heard', 'the', 'birds', 'chirping', 'loudly', 'in', 'the', 'morning']
Labels:    ['PRON', 'V', 'DET', 'NN', 'V', 'ADV', 'IN', 'DET', 'NN']
Predicted: ['IN', 'ADV', 'DET', 'NN', 'IN', 'ADV', 'IN', 'DET', 'NN']
```

Figure 4: Prediction and true POS labeling of LSTM



```
Sentence:  ['We', 'heard', 'the', 'birds', 'chirping', 'loudly', 'in', 'the', 'morning']
Labels:    ['PRON', 'V', 'DET', 'NN', 'V', 'ADV', 'IN', 'DET', 'NN']
Predicted: ['PRON', 'NN', 'DET', 'V', 'ADV', 'ADV', 'IN', 'DET', 'IN']
```

Figure 5: Prediction and true POS labeling of QLSTM

2.2 IMDB sentiment analysis

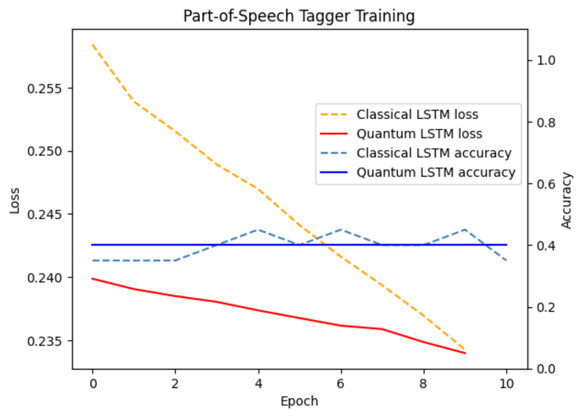


Figure 6: Loss and Accuracy curves for LSTM and QLSTM

3 Conclusion

- In the POS graph we can see that because the Quantum Computers are noisy after almost 50-55 epochs as training time keeps on increasing there are huge fluctuations
- On the other hand using a Quantum LSTM clearly starts us off at a better position than the classical LSTM as is shown by the loss curves
- we can see that on the IMDB dataset the accuracy of classical and Quantum both average out at the same values and the Quantum curve is also more smooth
- The time taken by the QLSTM model is much more than the classical one due to the overhead expense of the quantum circuit Due to this reason, we could only run for a small subset of the IMDB dataset and for a few epochs thus we can see low accuracy of just 40
- we can conclude here that Quantum models can be used only for specific tasks for all others their counterparts i.e. classical are much more

4 References

1. `https://arxiv.org/pdf/2009.01783.pdf`
2. `https://github.com/rdisipio/qlstm/tree/main#readme`
3. `https://towardsdatascience.com/sentiment-analysis-using-lstm-step-by`
4. `https://pennylane.ai/qml/demos/tutorial_kernel_based_training.html`