Maximum Independent Set

Hybrid quantum classical Neural Network

Edoardo Zappia

Deep Learning exam 31st July 2024

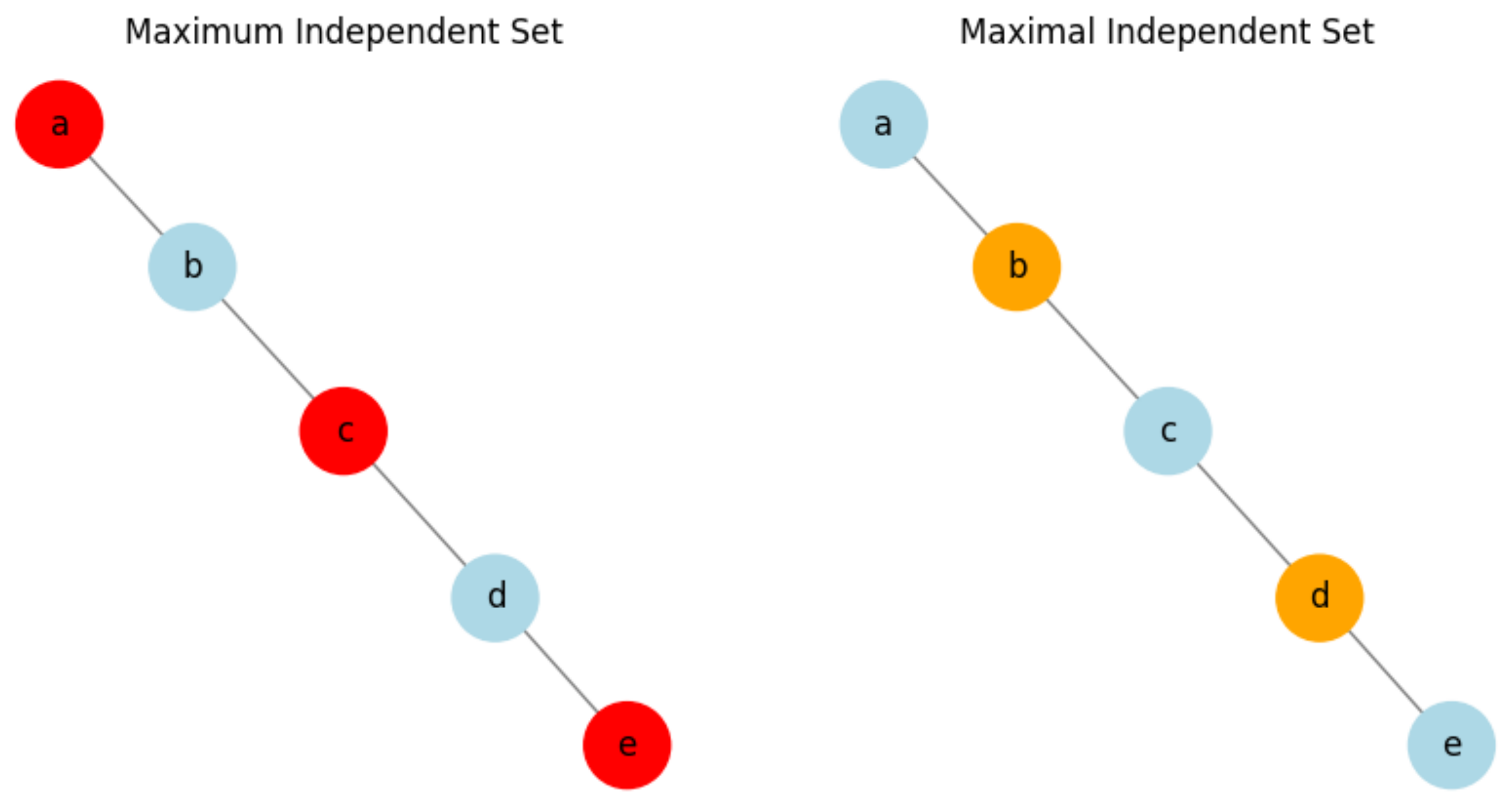
Maximum Independent Set

Problem definition

Given a graph $G = (V, E)$, where V is a set of vertices and E is a set of edges, the objective is to find the largest subset $I \subseteq V$ such that no two vertices in I are adjacent, i.e. for every pair of vertices $u, v \in I$, there is no edge $(u, v) \in E$. The size of the maximum independent set is the maximum number of vertices in such a subset.

Complexity class: NP-Hard

Applications: Wireless Networks, Social Networks, Graph Theory, Bioinformatics, Resource Allocation, VLSI Design, Computer Vision



Hybrid quantum classical NN

The whole system

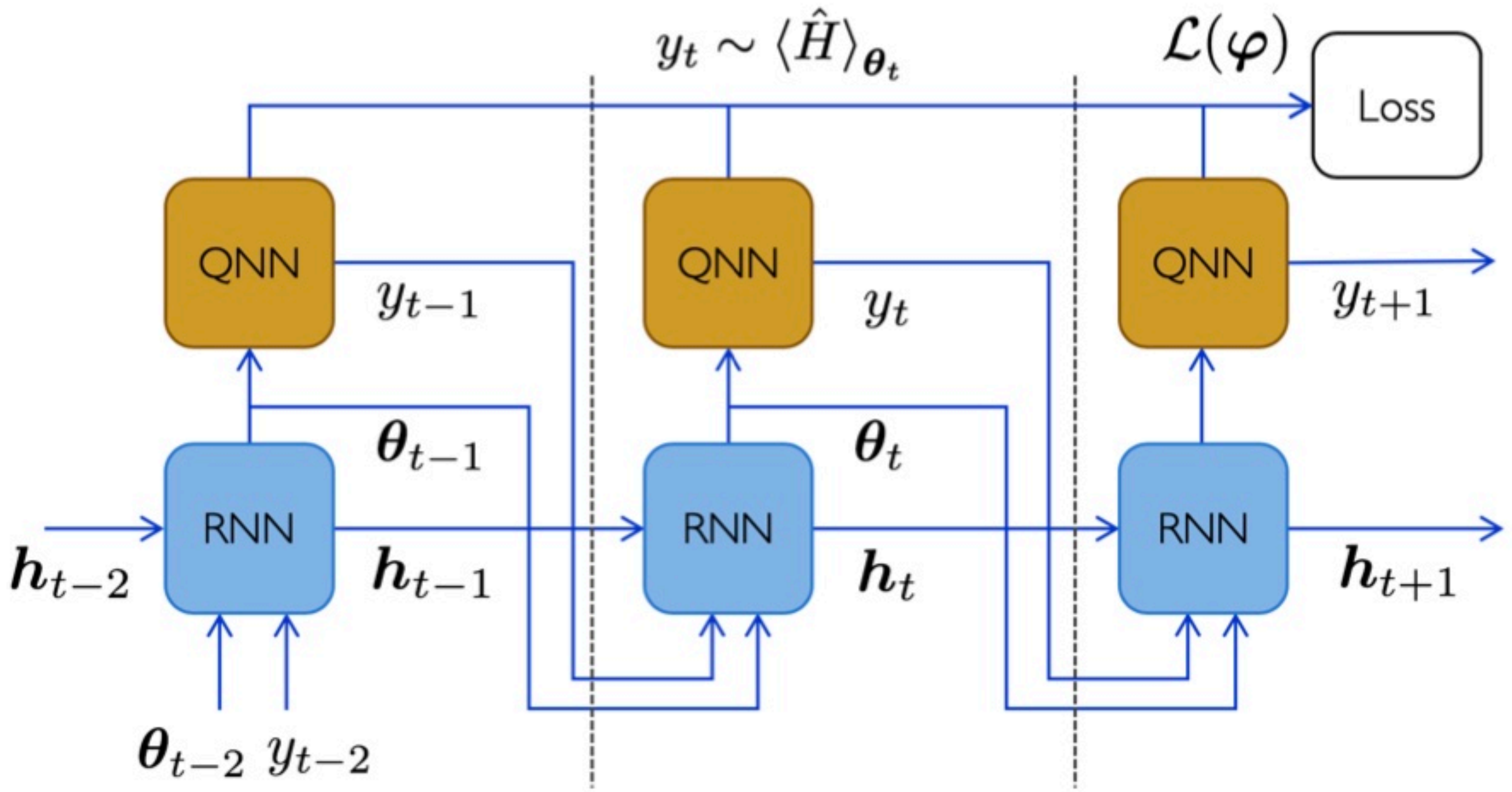
Tools

Quantum Neural Network (QAOA)

Recurrent Neural Network (LSTM or GRU)

Idea

To teach to the recurrent neural network to learn the weights of the quantum neural network



Hybrid quantum classical NN

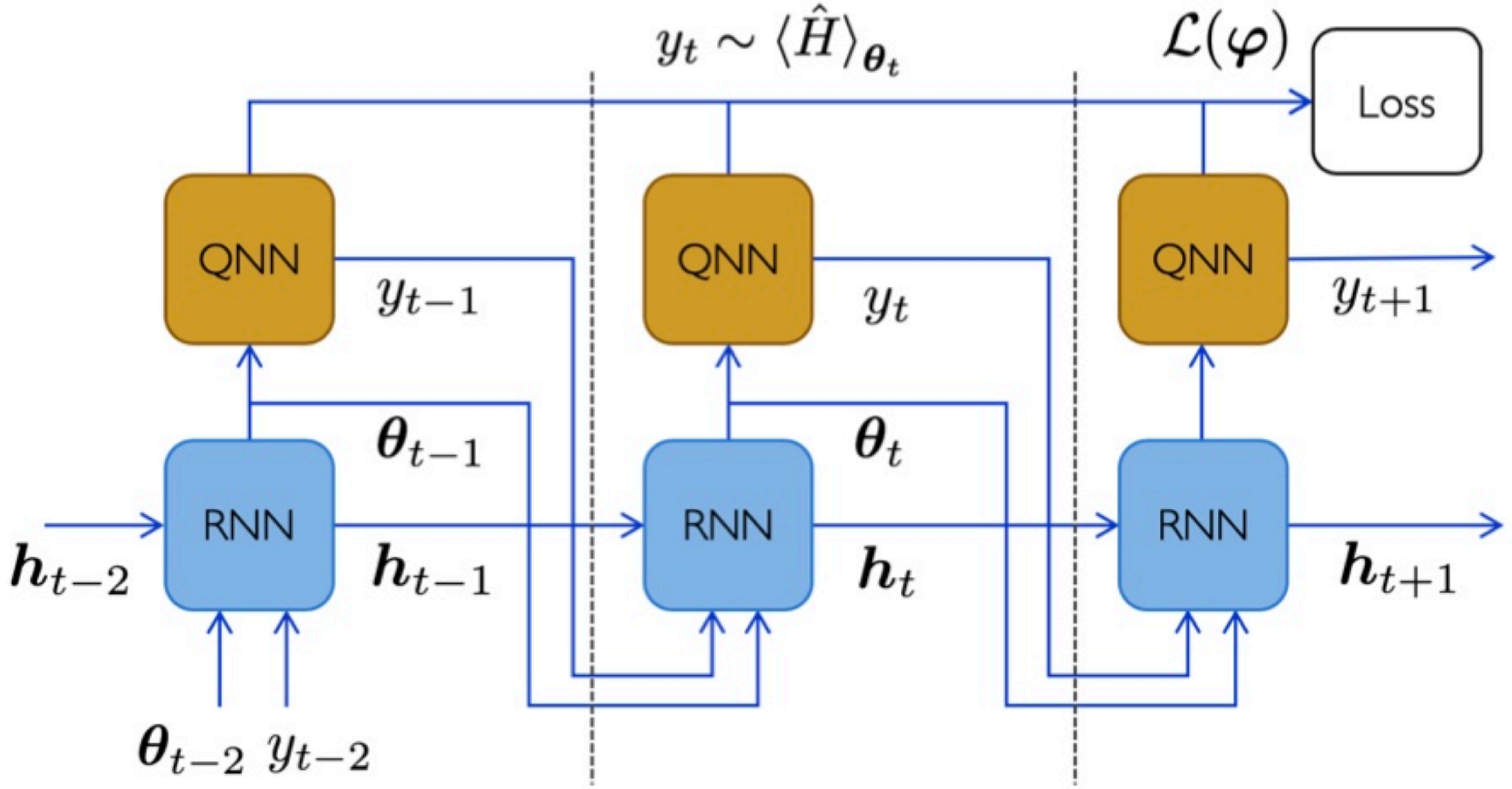
The whole system

Goal: To optimize the trainable parameters ϕ of the RNN with which it learns effective strategies for proposing optimal quantum circuit parameters

Iteration t^{th}

Recipients: parameters θ_t, cost y_t estimated as $\langle \hat{H} \rangle_{\theta_t}$, prior internal state h_t

Computation: $h_{t+1}, \theta_{t+1} = \text{RNN}_{\phi}(h_t, \theta_t, y_t)$



Quantum Approximate Optimization Algorithm (QAOA)

Quantum part (QAOA applied to MIS)

Configuration space:

The set of n-bit strings $x = x_1 x_2 \ldots x_n$ representing V^* where $x_i = 1 \iff i \in V^*$

Objective function:

$$f(x) = \sum_{j=1}^{n} x_j$$ where $f(x)$ is the number of vertices in V^*

Phase Hamiltonian:

$$H_P = \sum_{u \in V} \frac{1}{2}(I - Z_u) = \frac{n}{2} I - \frac{1}{2} \sum_{u \in V} Z_u$$

Phase Operator:

$$U_P(\gamma) = e^{i\frac{\gamma}{2}\sum_{u \in V} Z_u} = \sum_{u \in V} e^{i\frac{\gamma}{2} Z_u}$$

Mixing Hamiltonian:

$$H_{M,u} = \frac{1}{2^\alpha} X_u \prod_{j=1}^{\alpha} (I + Z_{v_j})$$

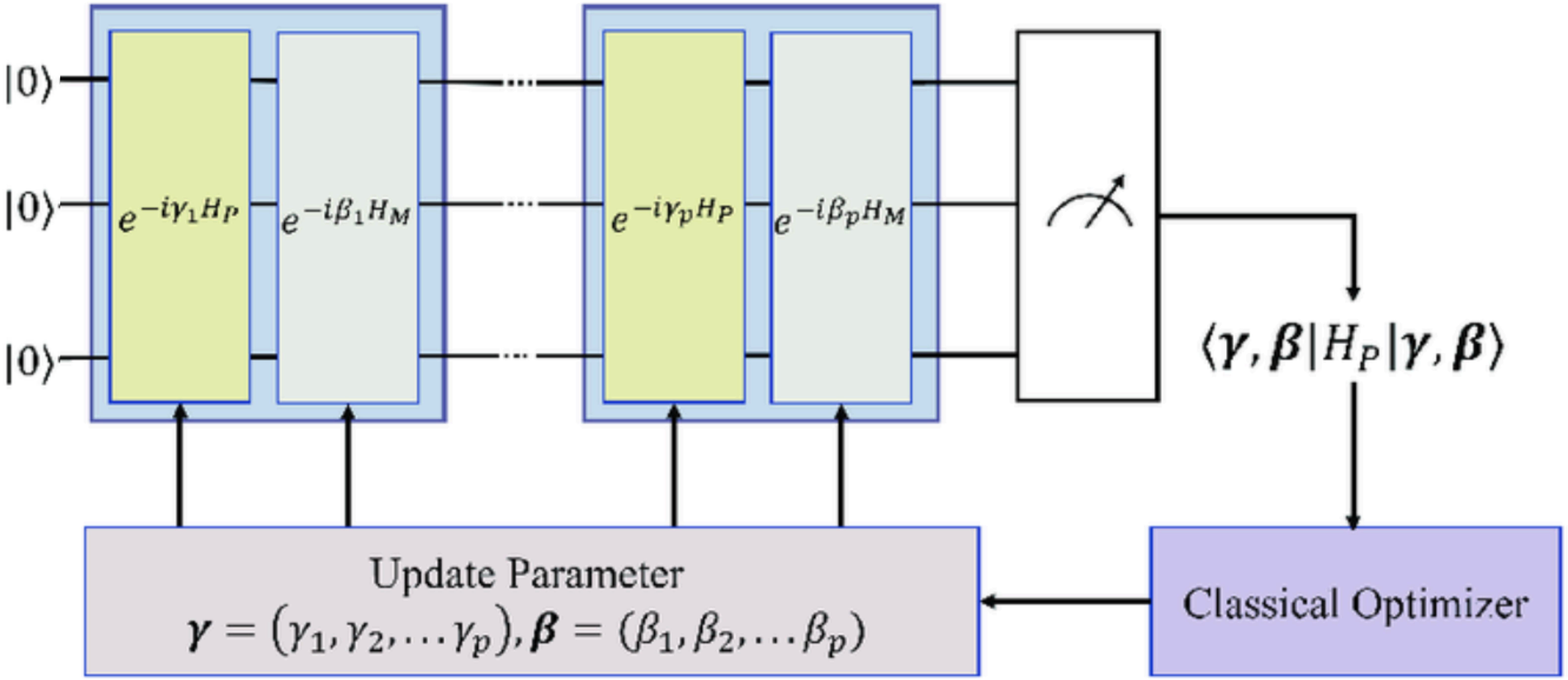
Mixing Operator:

$$U_{M,u}(\beta) = e^{-i\beta H_{M,u}}$$

Quantum Approximate Optimization Algorithm (QAOA)

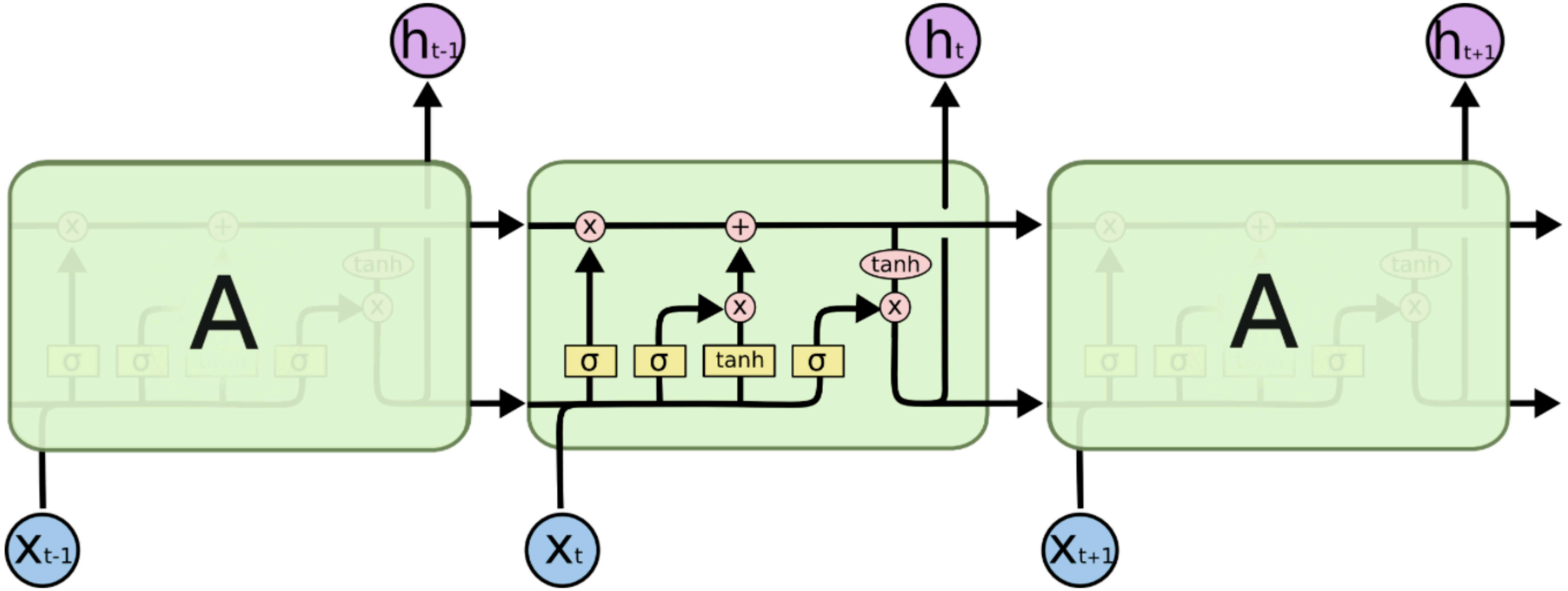
Quantum part

Circuit: $U(\gamma, \beta) = e^{-i\gamma_p H_P} e^{-i\beta_p H_m} \ldots e^{-i\gamma_1 H_P} e^{-i\beta_1 H_M}$



Recurrent Neural Network

Long Short Term Memory (LSTM)

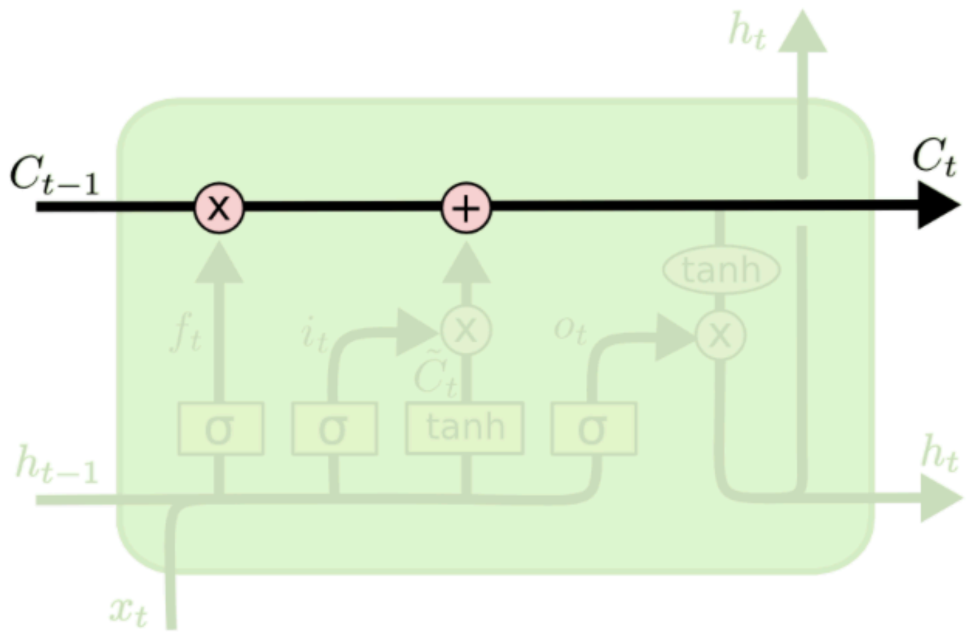


Long Short Term Memory networks are a special kind of RNN, capable of learning long-term dependencies

Recurrent Neural Network

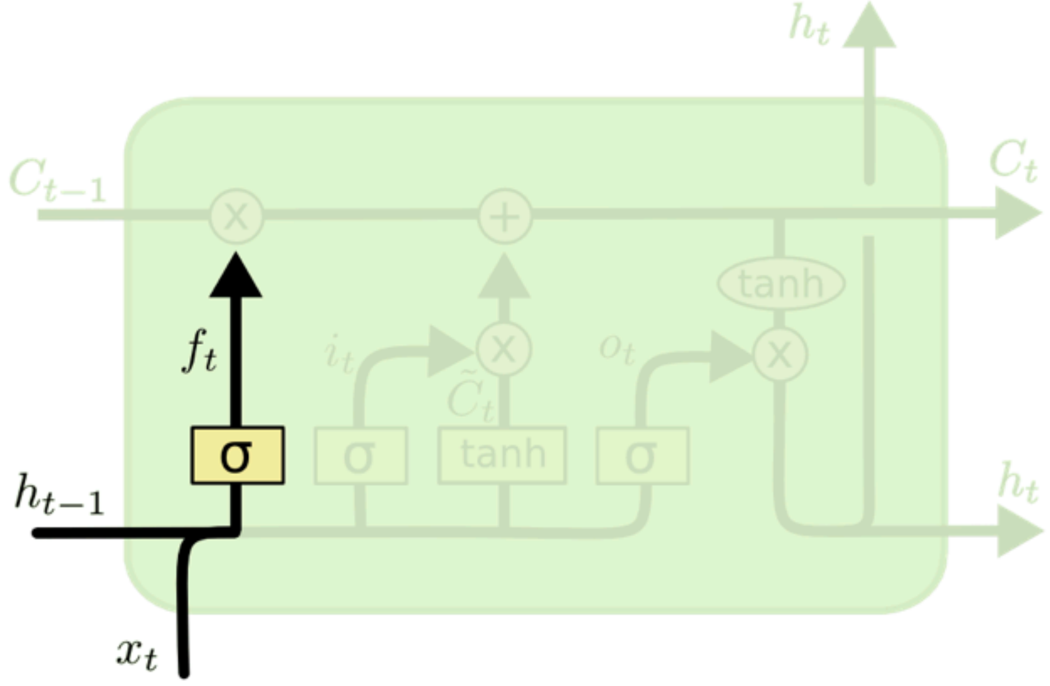
Long Short Term Memory (LSTM)

Cell State:



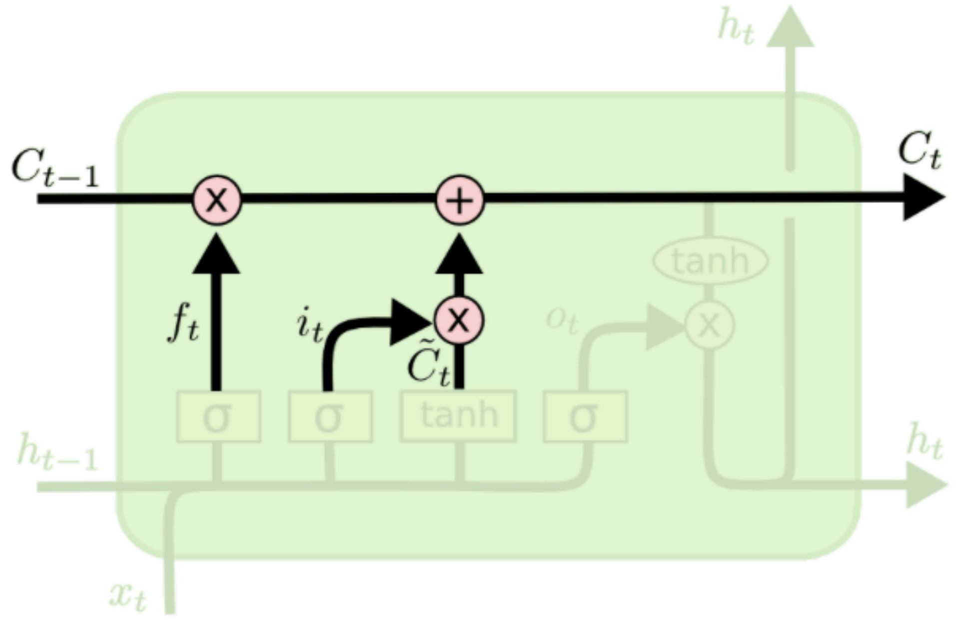
Forget gate layer:

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

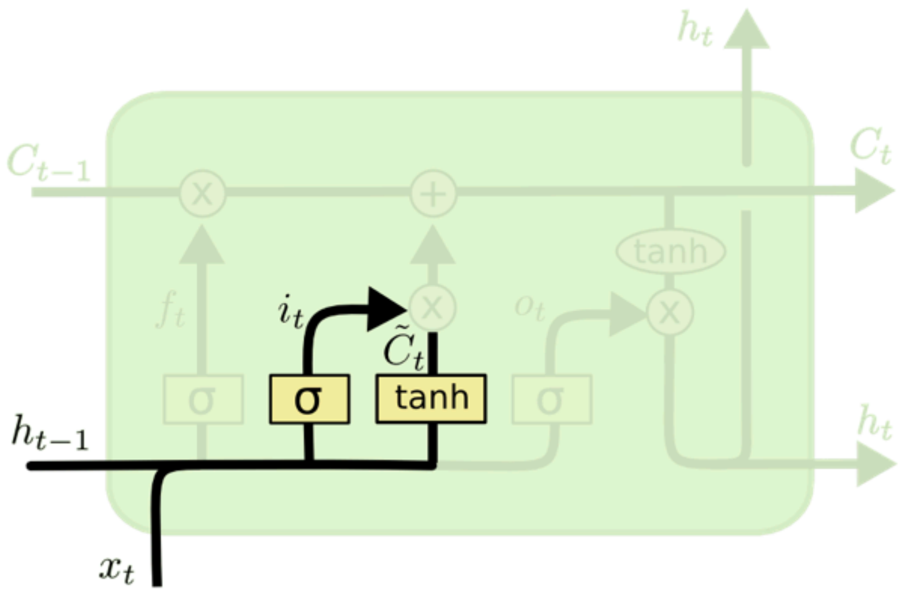


Just do it:

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t$$



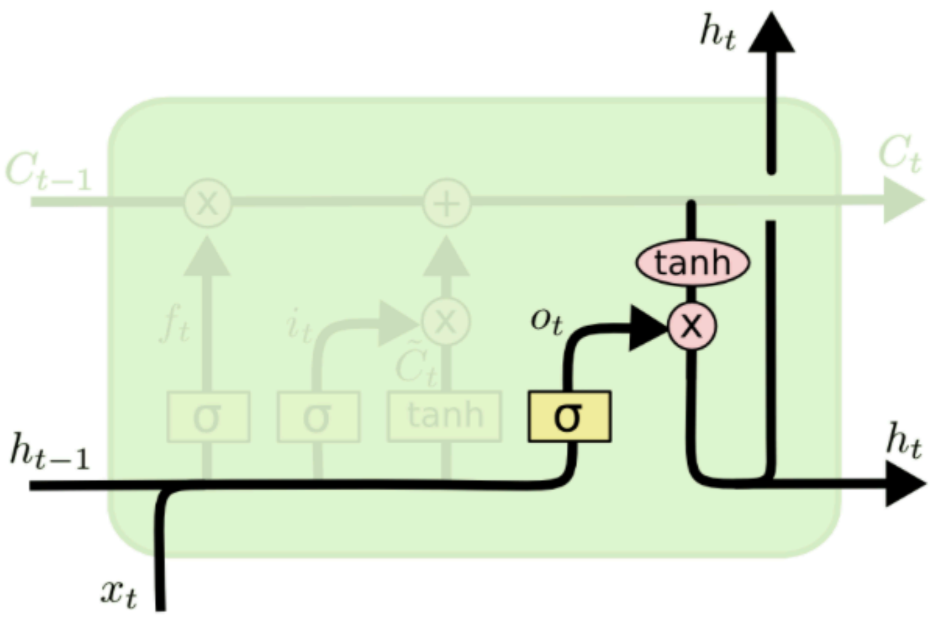
Input gate layer and new candidate values:



$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + bi)$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Output:



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

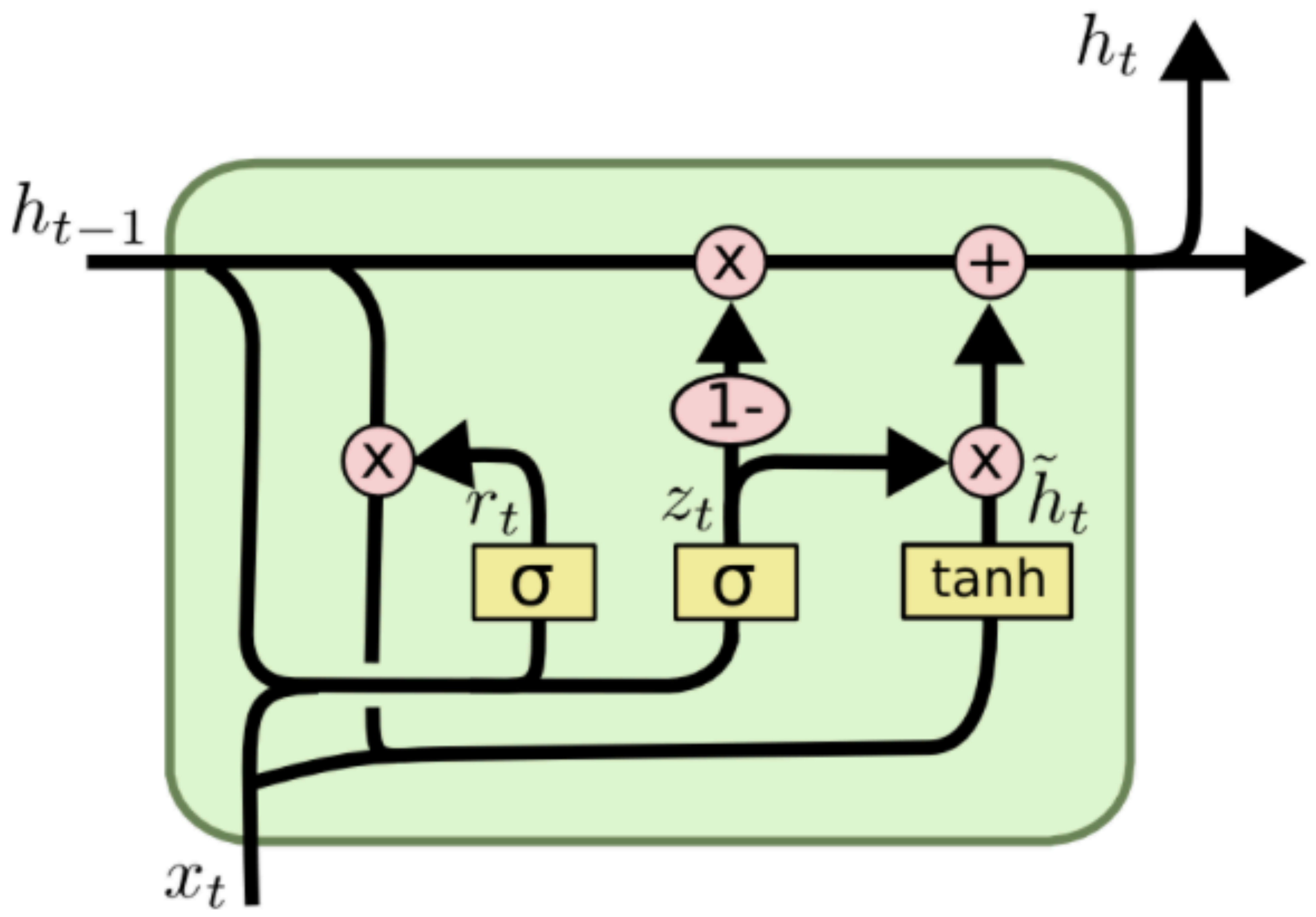
$$h_t = o_t \times tanh(C_t)$$

Recurrent Neural Network

Gate Recurrent Unit (GRU)

State: merges the cell state and hidden state

Update gate: combines the forget and input gates



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

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

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

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

Experimental setup

Technical details

QAOA layers: 2

LSTM Cells: 4

GRU Cells: 4

Train dataset: graphs with

$n_{nodes} \in [3,6]$

$k \in [2, n_{nodes} - 1]$

$edge\_prob = \frac{k}{n_{nodes}}$

Test dataset:

one random graph with

$n_{nodes} = 12$

$edge\_prob = \frac{3}{7}$

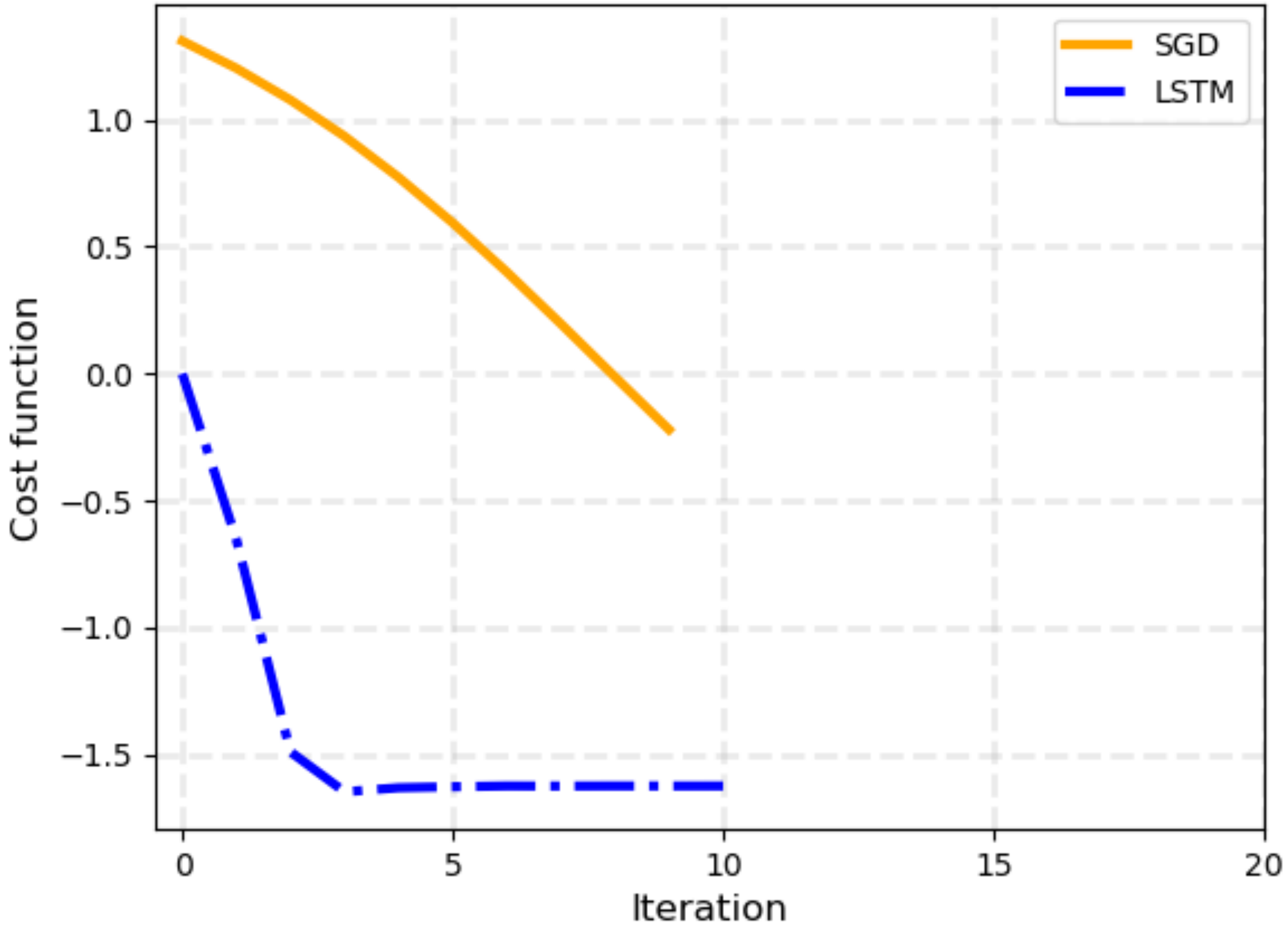
Quantum part: Pennylane

Classical part: Tensorflow

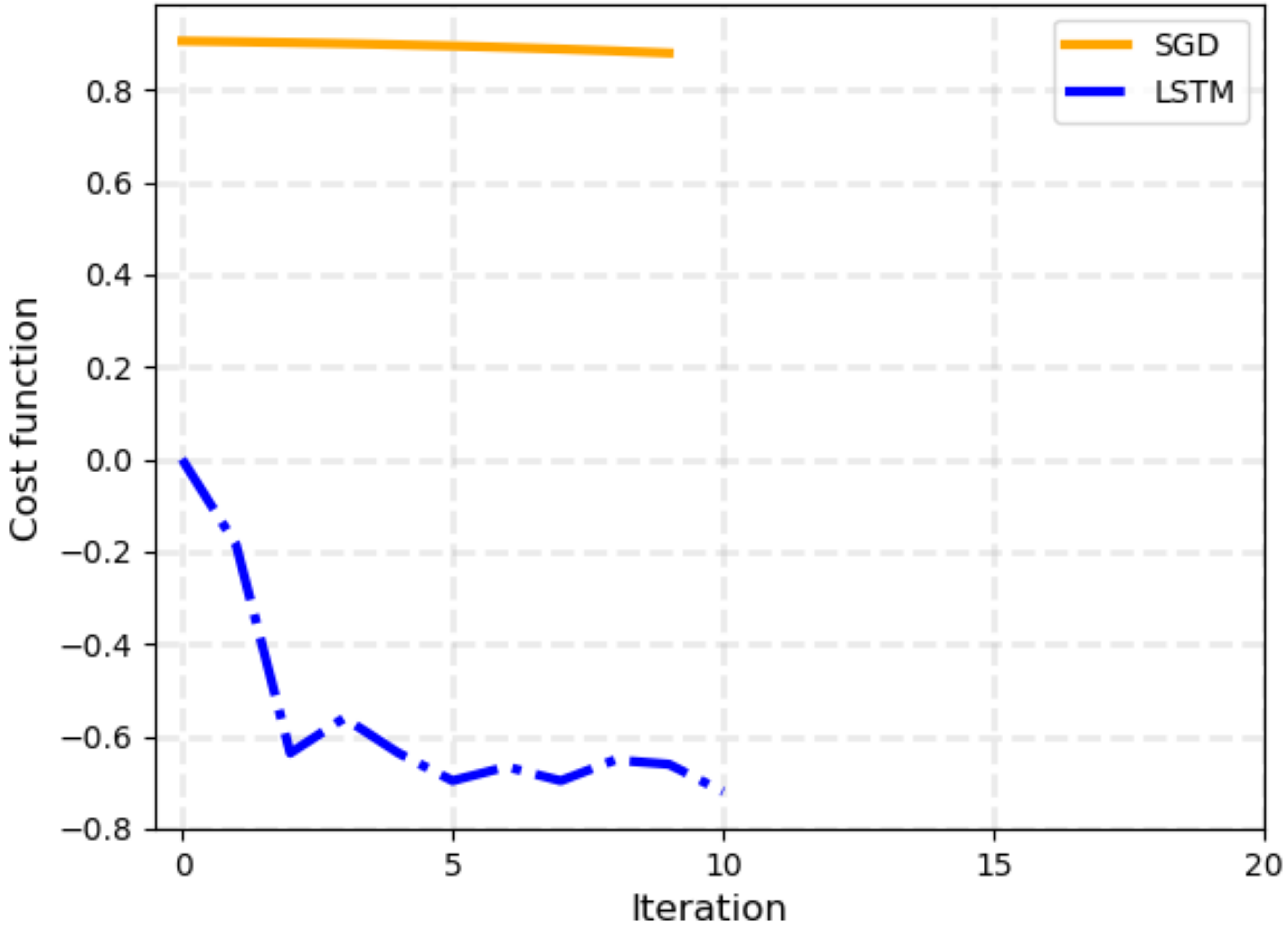
Graphs: Networkx

Results

LSTM vs SGD

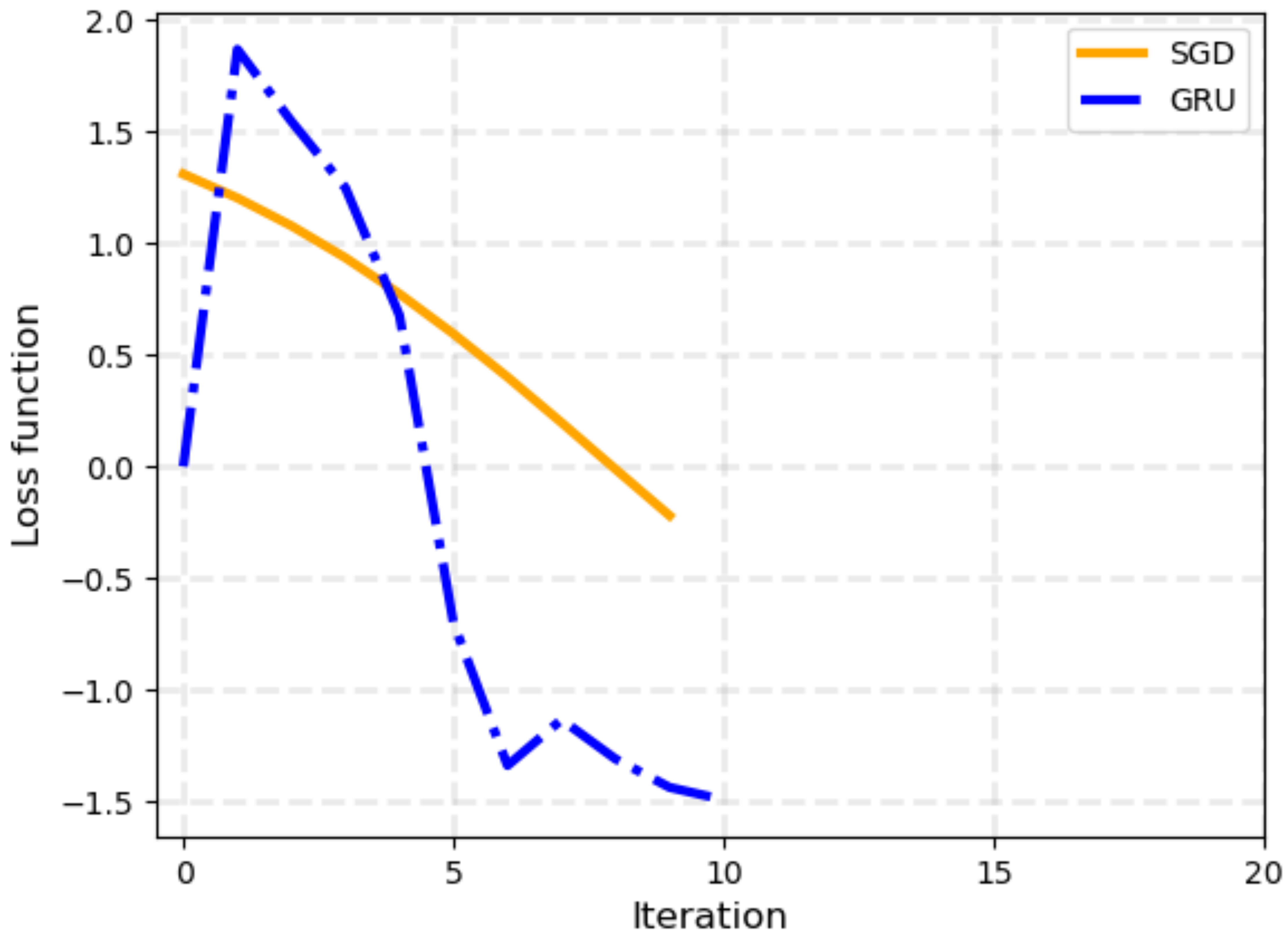


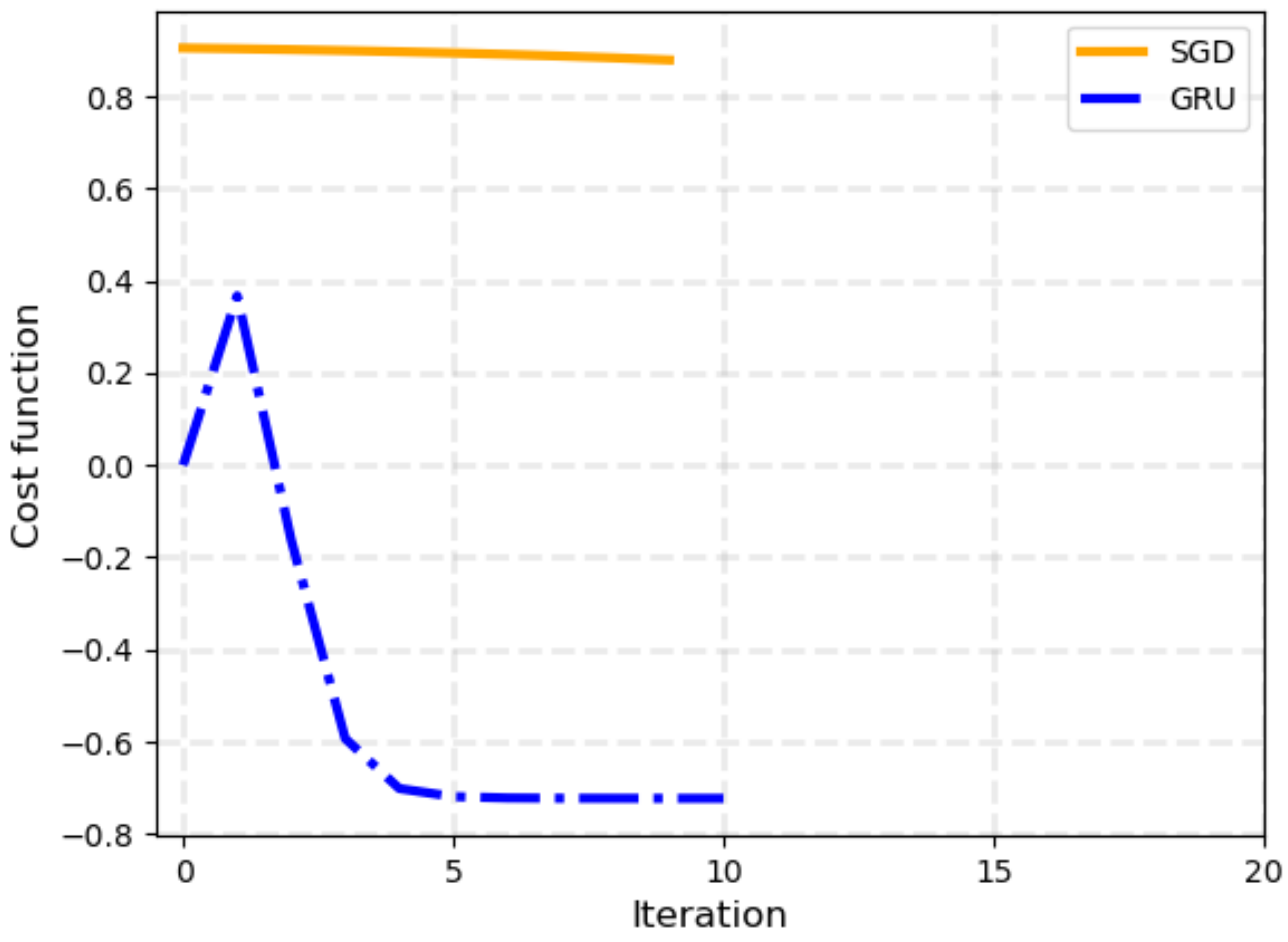
Dataset: 40 graphs, 10 epochs



Dataset: 120 graphs, 40 graphs per batch, 10 epochs

Results
GRU vs SGD





Dataset: 40 graphs, 10 epochs

Dataset: 120 graphs, 40 graphs per batch, 10 epochs

Conclusions

Pros:

Generalization

Lower number of iterations

Better results

Cons:

Slow training

Computational cost

Optimizing the parameters of a quantum circuit is complex, even with Stochastic Gradient Descent (SGD). The system used requires slow training, but it saves time and resources during the testing phase on larger graphs. In fact, better minima are achieved in very few iterations.