(24743) - A QISKIT FRAMEWORK FOR QLSTM TO PREDICT REAL WORLD TIME SERIES DATA

Stefan Kister (Germany)[1*]; Jonas Michel (Germany)[2]; Felix Lehner (Germany)[3]

[1] ParTec AG. [2] Master Student (University of Galway) [3] IBM Deutschland GmbH *Qiskit Advocat

Abstract:

Long Short-Term Memory (LSTM) methods were introduced to overcome the vanishing gradient problem of recurrent neural networks in the 90's. LSTM is still valuable method in deep learning and got recently enhanced by an extended LSTM method. LSTM can be used in different use cases, e.g. language modeling, machine translation (sequence to sequence learning), image captioning, handwriting generation, image generation using attention models.

On basis of first published approaches of quantum LSTM we introduce a qLSTM framework utilizing the integrated PyTorch workflow in Qiskit. In our demonstrator we use LSTM for predicting time series data in a hybrid quantum-classical workflow. We use variational quantum circuits (VQA) for representing the weights in the classical LSTM cell. The variational quantum circuits are built of 2 parts: An encoding layer (quantum feature map) to load classical data points into quantum feature space, and a variational layer (ansatz) which contains the tunable parameters.

We will show illustrative examples utilizing the Aer state vector simulator for noise free QC simulations as well as fake backends with noise models from current IBM eagle devices. To illustrate functionality of the qLSTM framework results will be presented with toy data of a damped oscillator, but also with real world data which were used to create climate stripes and financial transaction predictions.

Introduction

Long Short-Term Memory (LSTM) has been introduced by Hochreiter and Schmidthuber in the 90th of last century [1,2,3]. It has been used in various domains and could demonstrated a high effectiveness at numerous sequence-related tasks. It has been developed to overcome the vanishing gradient problem of recurrent neural networks (rNN) by using a constant error carousel and gating [1,4]. Although LSTM is a long-established method, this method still offers great utility and usage, which was highlighted by the recent publication of the extended LSTM method [5].

The ML workflow of LSTM is shown in Fig. 1(a). Data is transformed from model representation to a multi-dimensional tensor, see Fig 1(b), the shape of the input and target tensors depend on the number of time steps and the machine learning step. Pre-processed data flows into a network architecture which is represented by an LSTM cell, see. Fig. 1(c). The RNN is trained with the input data and then evaluated with a feedback loop to optimize set of hyper parameter, see Fig. 1(d). The optimization is gradient-based, and the evaluation criteria are „Mean Squared Error (MSE)" and „Mean Absolute Error (MAE)".

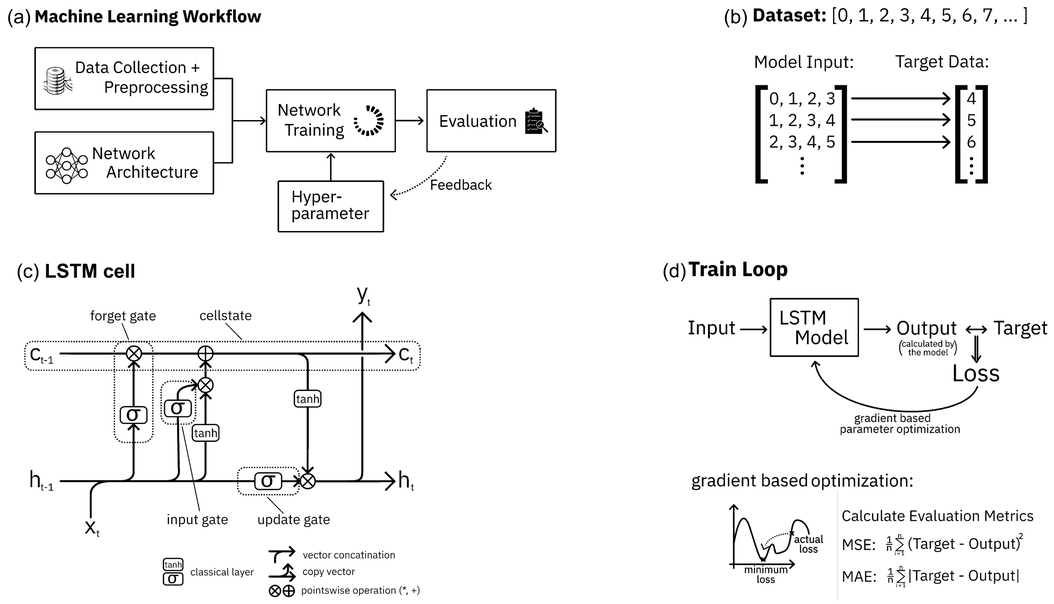

Fig. 1: (a) ML workflow, (b) Dataset transformation model matrix to target vector, (c) Network architectur – classical LSTM cell [6], (d) LSTM training and evaluation procedure

The gates and cell states of the LSTM cell can be described with the following equitations (1)-(6):

Forget Gate determines what information from the previous cell state C(t-1) should be discarded:
(1) $f(t) = \sigma(W_f * [h(t-1), x(t)] + b_f)$

Input Gate controls which values from the input x(t) and the previous cell state C(t-1) should be updated:
(2) $i(t) = \sigma(W_i * [h(t-1), x(t)] + b_i)$

Candidate Cell State based on the input x(t) and the previous hidden state h(t-1):
(3) $C(t) = \tanh(W_C * [h(t-1), x(t)] + b_C)$

Update Cell State is based on the forget gate, input gate, and candidate cell state:
(4) $C(t) = f(t) * C(t-1) + i(t) * C(t)$

Output Gate determines the next hidden state h(t) based on the current cell state:
(5) $o(t) = \sigma(W_o * [h(t-1), x(t)] + b_o)$

Final Hidden State based on the cell state and the output gate :
(6) $C(t) = o(t) * \tanh(C(t))$

qLSTM Approach

Since the first ideas on quantum LSTM were published in 2020 [6], the topic has developed into an emerging research field in various areas., for example in material synthesis [7], source code analysis [8], NLP [9], GAN [14], or time series prediction [11, 16, 18]. Studies comparing classical LSTM [11, 16] and extensions of qLSTM [10, 12, 13, 15, 17] have also been published. The promises of qLSTM depends on the applications, for example in some of the mentioned cases higher accuracy has been achieved or a lower dimensional hyperparameter space was necessary to achieve same quality in results.

In principle the qLSTM methods follows the classical LSTM workflow (Fig. 2(a), using the same LSTM cell, but instead of classical weights quantum variational circuits (VQC) are used, see Fig. 2 (a). A VQC consists of 2 layers: (1) a data encoding layer represented by quantum feature map which encodes data features in to a quantum hilbert space (fixed parameter) (2) a variational layer represented by an ansatz which is a parametrized quantum circuit. Both faeture map and ansatz are configurable, a good overview is given e.g. in [19].

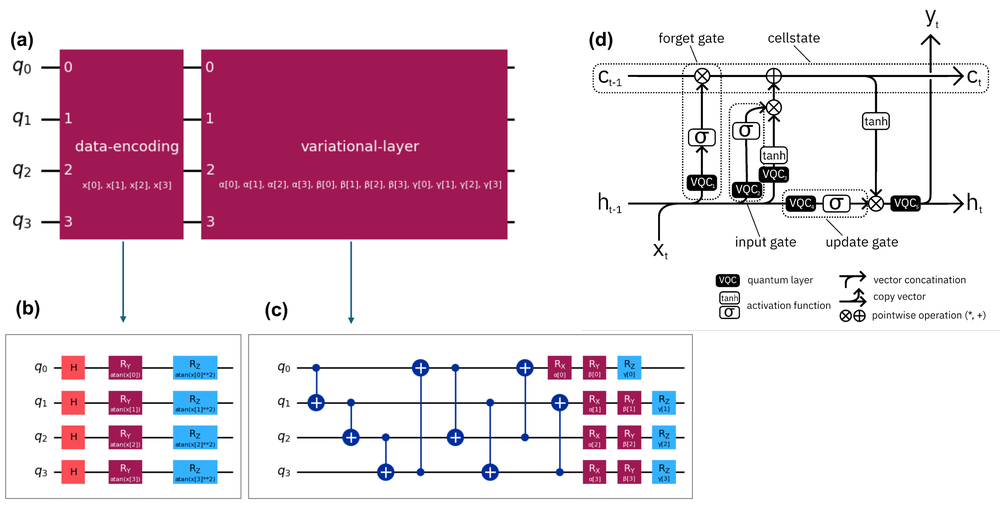

Fig. 2: (a) VQC with data encoding and variational layer, Illustrative examples of (b) quantum feature map as data encoding layer and (c) ansatz as variational layer (used in [6]), (d) qLSTM cell with VQCs as weights [6]

These parameters are optimized in a hybrid quantum-classical execution loop according to the variational principle. For this optimization, a minimum of a Hamiltonian operator is computed. The quantum computer is used to measure a wave function represented by a parameterized quantum circuit. The measurement results are then fed into a classical optimizer to obtain a new set of parameters for the next measurements on the quantum computer until convergence is achieved. The Hamiltonian operator is in principle configurable. In this poster work, a Hamiltonian operator common in QML was used, a tensor product of Z-Pauli operators (7):

(7) $$H = c \prod_{i=1}^{n} Z_i \qquad with\ n = number\ of\ qubits \equiv feature\ dimension$$

qLSTM Workflow and Qiskit framework

The Qiskit framework for qLSTM (see Fig. 3) was developed in the form of generator files that were programmed in Python along the elements of a qLSTM workflow. The data preparation must be adapted to the respective data set. The data then flows into the qLSTM initialization, which is followed by model training. After evaluation, the quality of the model predictions is determined via the losses. The initialization of the qLSTM is supplemented by corresponding generator files programmed in Qiskit. Here, different feature maps, approaches, noise models and simulated or real quantum devices can be selected as the backend, or your own elements can be added. An qLSTM prediction of a certain data set can be parameterized in and then executed by qLSTM.py file. The framework will be made available soon as open source (work in progress).

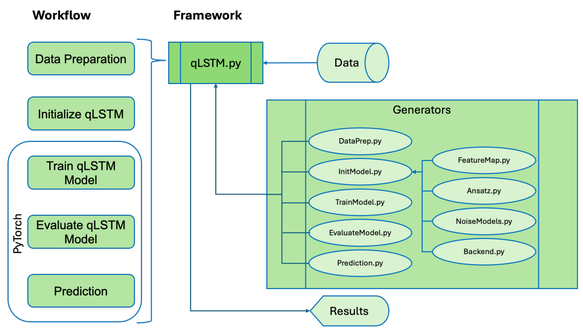

Fig. 3: qLSTM workflow and Qiskit framework

Results

Example 1: Damped Oscillator [20]

The Damped Oscillator is utilized as a toy dataset to proof the feasibility of the QLSTM [6]. Figure 4 shows a Comparison of 3 proposed circuits in the literature [6, 21, 22]. The comparison was run with the framework code to ensure the comparability. Figure 4 highlights the importance of the chosen quantum circuit, since one qLSTM gets outperformed by the other ones. Figure 5 shows the predicted data in respect to the trainings data over the Epochs. The red line indicated the train-test split, where the test split was never seen by the model during training.

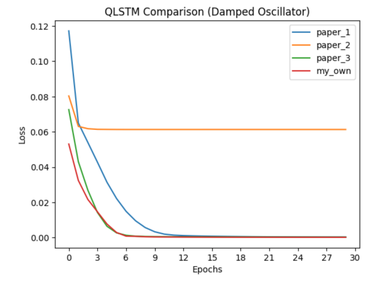

Fig. 4: qLSTM Circuit Comparison

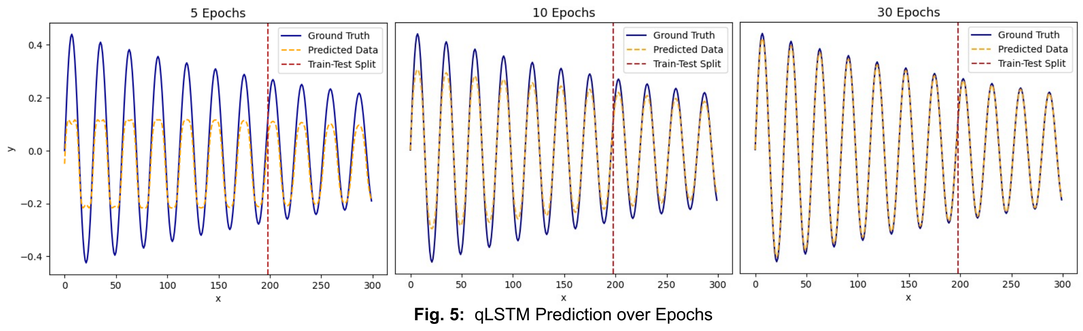

Fig. 5: qLSTM Prediction over Epochs

Example 2: Financial Transaction Prediction [19]

Dataset Overview

For benchmarking the forecasting capabilities of our hybrid qLSTM model, we utilized the 1999 Czech Financial Dataset, which consists of real anonymized credit card transactions released for the PKDD'99 Discovery Challenge. The dataset contains financial records from Czech banks, including account information, loan records, and transaction details, covering the period from 1993 to 1998.

The dataset is divided into 9 files, each containing specific financial information. Our focus is on the Transactions file, which holds data on credit card transactions, as we want to predict the amount of the transaction. The used data is the difference of the transactions amounts, which is seen in Fig. 6.

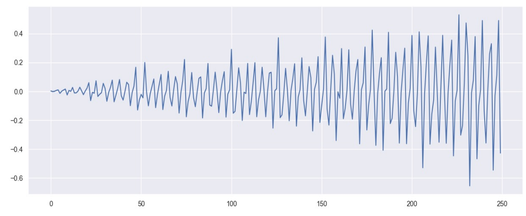

Fig. 6: Normalized difference in Transaction amount over time in the bank account

Final Model Setup

VQC - Feature Map:

VQC – Ansatz:

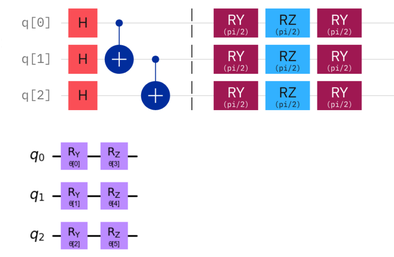

Fig. 7: Final quantum circuits for data encoding (Feature Map) and variational layer (Ansatz).

Results

The table Tab. 1 shows various LSTM models based on performance metrics: parameters, average Mean Absolute Error (MAE), Mean Squared Error (MSE), and computational time.

The LSTM PyTorch X model achieves the best performance, with an MAE of 0.1878, an MSE of 0.0673, and a runtime of 0.042 minutes. In contrast, QLSTM #10 offers a competitive MAE of 0.2087 and an MSE of 0.0769 but has a longer runtime of 55 minutes. Other QLSTM models show a trade-off between accuracy and efficiency.

Figures 8and 9 illustrate predictions from the classical QLSTM and LSTM models, respectively. Both align closely with the actual time series data, but the QLSTM shows smoother oscillation handling, particularly at the end, indicating potential for capturing long-range dependencies.

Model	# of Parameters	Avg. MAE	Avg. MSE	Time in min
LSTM Keras	20	0.2568	0.1237	0.067
LSTM PyTorch	24	0.1920	**0.0658**	0.028
LSTM PyTorch X	20	**0.1878**	0.0673	0.042
QLSTM #10	24	**0.2087**	**0.0769**	55
QLSTM #999	12	0.2376	0.0878	40
QLSTM #4	72	0.2318	0.0961	90

Tab. 1: Comparison of LSTM vs QLSTM with different settings and enviroments.

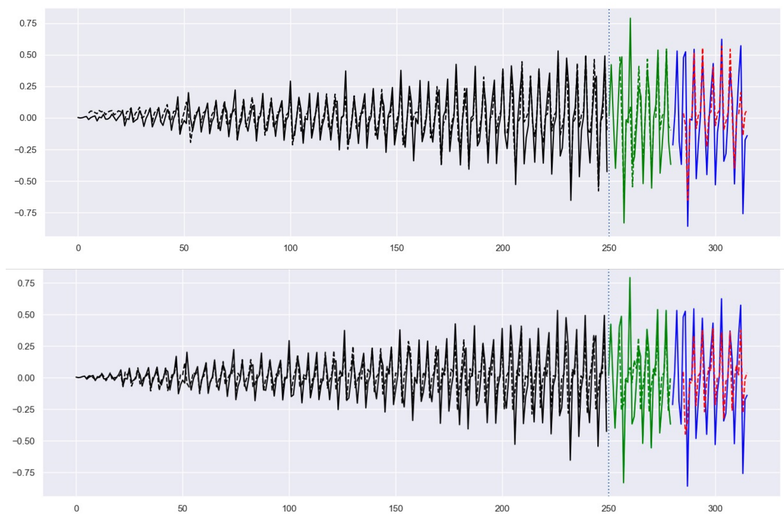

Fig. 8: Predictions (dashed) from the QLSTM model

Fig. 9: Predictions (dashed) from the classical LSTM model

Example 3: Climate stripes prediction (work in progress)

Climate stripes showing intuitively the global warming is presented every by Ed Hawkins from University of Reading (UK) [23]. The last publish stripes are shown in Figure 10(a). As a further example of predicting a time series based on real data with qLSTM we chose the climate data give by MetOffice [24] for the global temperature change from 1850-2018. As a first approach we predicted the missing temperature data fdata for 2019-2023 (see Figure 10 (b)). Further improvements of the qLSTM model and prediction to a further future are currently work in progress and the complete results are planned to be published soon.

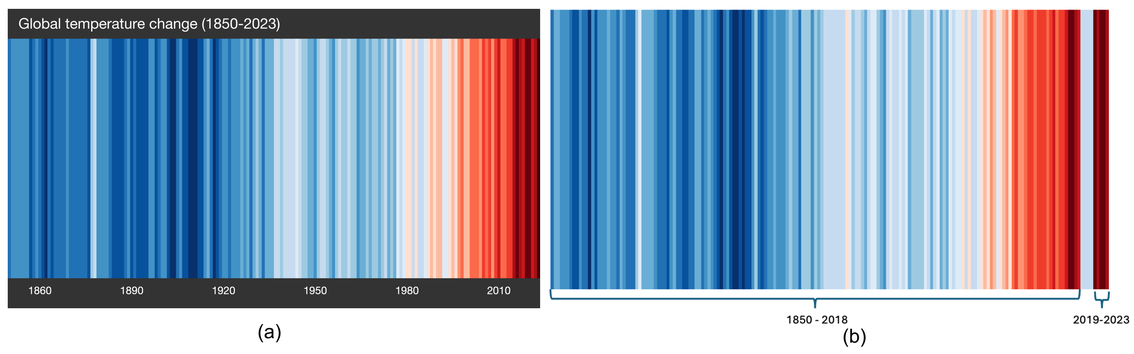

Fig. 10: Climate Stripes (a) real data provided by [23] 1850-2023 (b) Climate Stripes unsed for qLSTM Training 1850-2018 and predicted results for 2019-2023.

References:

[1] S. Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. Master's thesis, Technische Universität München, 1991.
[2] S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural Computation, 9(8):1735–1780, 1997a. S.
[3] Hochreiter and J. Schmidhuber. LSTM can solve hard long time lag problems. In M. C. Mozer, M. I. Jordan, and T. Petsche (eds.), Advances in Neural Information Processing Systems (NeurIPS), volume 9, pp. 473–479. MIT Press, Cambridge MA, 1997b.
[4] S. Hochreiter, et al.. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In J. Kolen and S. Kremer (eds.), A Field Guide to Dynamical Recurrent Networks. IEEE, 2000.
[5] Beck et al., "xLSTM.", https://doi.org/10.48550/arXiv.2405.04517
[6] Chen, Yoo, and Fang, "Quantum Long Short-Term Memory.", https://doi.org/10.48550/ARXIV.2009.01783
[7] Beaudoin et al., "Quantum Machine Learning for Material Synthesis and Hardware Security.", https://doi.org/10.48550/arXiv.2208.08273
[8] Akter, Mst Shapna, Hossain Shahriar, and Zakirul Alam Bhuiya. "Automated Vulnerability Detection in Source Code Using Quantum Natural Language Processing." arXiv, March 13, 2023. https://doi.org/10.48550/arXiv.2303.07525.
[9] Stein et al., "Applying ONLP to Sentiment Analysis in Finance.", https://doi.org/10.1109/ICCE57702.2023.10178
[10] Chen, "Efficient Quantum Recurrent Reinforcement Learning via Quantum Reservoir Computing.", https://doi.org/10.48550/arXiv.2309.07339
[11] Khan et al., "Quantum Long Short-Term Memory (QLSTM) vs Classical LSTM in Time Series Forecasting.", https://doi.org/10.48550/arXiv.2310.17032
[12] Chehimi et al., "Federated Quantum Long Short-Term Memory (FedQLSTM).", https://doi.org/10.48550/arXiv.2312.14309
[13] Chen, "Learning to Program Variational Quantum Circuits with Fast Weights.", https://doi.org/10.48550/arXiv.2402.17760
[14] Chu, Hastak, and Chen, "LSTM-QGAN.", https://doi.org/10.48550/arXiv.2409.02212
[15] Zhou et al., "Implementation Guidelines and Innovations in Quantum LSTM Networks.", https://doi.org/10.48550/arXiv.2406.08982
[16] Mahmood et al., "Comparative Study of Long Short-Term Memory (LSTM) and Quantum Long Short-Term Memory (QLSTM).", https://doi.org/10.48550/arXiv.2409.08297
[17] Chen et al., "Reservoir Computing via Quantum Recurrent Neural Networks.", https://doi.org/10.48550/arXiv.2211.02612
[18] Lin, Liu, and Chen, "Quantum-Train Long Short-Term Memory.", https://doi.org/10.48550/arXiv.2407.08617
[19] Felix Lehner, Leveraging Quantum enhanced Long short-term memory networks in analyzing time series", Master Thesis Department VI – Informatik und Medien – Data Science – Machine Learning, BHT Berlin, 31. August 2023
[20] Jonas Michel, 2. Projektarbeit, Fakultät Wirtschaft und Gesundheit Studiengang Wirtschaftsinformatik - Kurs WWI2021F, DHBW Stuttgart, 20.11.2023
[21] Yu, Y. et al. (2023): Prediction of Solar Irradiance One Hour Ahead Based on Quantum Long Short-Term Memory Network. In: IEEE Transactions on Quantum Engineering 4, S. 1–15. doi: 10.1109/TQE.2023.3271362.
[22] Qi, J./Yang, C.-H. H./Chen, P.-Y. (2021): QTN-VQC: An End-to-End Learning framework for Quantum Neural Networks. arXiv: 2110.03861 [quant-ph].
[23] Edward Hawkins, #ShowYourStripes, https://showyourstripes.info
[24] HadCRUT4 Data, Met Office Hadley Centre observations datasets, Download: https://www.metoffice.gov.uk/hadobs/hadcrut4/data/current/download.html