Quantum Machine Learning Course Report

Applying Quantum Autoencoders for Time Series Anomaly Detection

Filippo Albani

Università degli Studi di Firenze

January 18, 2025

Contents

Introduction

Robin Frehner and Kurt Stockinger's article, "Applying Quantum Autoencoders for Time Series Anomaly Detection" [1], compares quantum autoencoding algorithms trained and tested on the UCR time series classification archive [2]. **Their experimental results demonstrate that, both with simulated and real quantum hardware, quantum autoencoders achieve superior anomaly detection performance while utilizing 60-230 times fewer parameters and requiring five times fewer training iterations**.

This report summarizes the key findings of the article and offers practical insights to help students and readers unfamiliar with the field better understand the research. It also provides guidance on recreating the simulations and experiments discussed. Given resource constraints, this report focuses primarily on the classical benchmarks and simulated quantum experiments, leaving real hardware experiments for future exploration.

The experiments reported in this work have been conducted on datasets not reportedly used in [1] but still from the same archive [2]. The notebooks used for this report are available in the associated GitHub repository: `https://github.com/filippoalbani/quantum_autoencoders`.

Preliminaries

In this section, the title of the article [1] will be broken down to provide the reader with an understanding of the starting point of this research: the type of data analyzed, the information sought, and the tools used. The **quantum** aspect, being the central focus of the article, is reserved for a dedicated chapter.

Time series

This work stems from the importance, in many fields, of the analysis of time series, which are collections of measurements of the same quantity at different (consecutive and even-spaced) times. The quantity could be a biometric value for medical diagnosis purposes, the amount of money transfers in fraud detection, or any other signal in almost every field of science.

In practice, we work with the UCR time series classification archive [2] which contains datasets of time series from diverse fields. Each dataset comes along with a read-me commentary file and is already parted in two .tsv files ready to use for machine learning purposes: one file for the training, containing no *anomalies* (defined below), and **one file for the test, containing one and only one anomaly**. The .tsv files show matrixes where the first element in the row is the class label (not relevant in this work). The rest of the row elements are the data sample values. The order of time series exemplar carry no special meaning and is in most cases random.

Anomaly detection

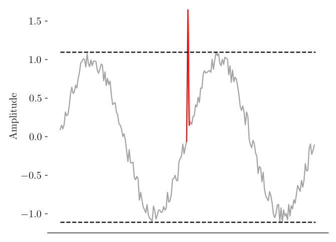


Figure 1: Point anomaly, where a value of 1.5 exceeds the normal range (dashed lines).

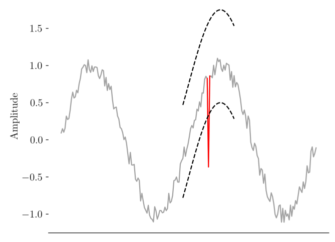


Figure 2: Contextual anomaly, where the value is typical but anomalous given its context.

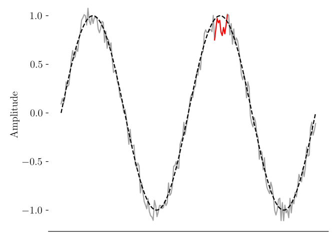


Figure 3: Collective anomaly, where data points are individually within normal noise levels but collectively exhibit a prolonged period of lower values.

We are often interested in fitting the time series to a function that describes its behavior, but in this study we are interested in another problem which is just as important: we want to build an algorithm which inputs a specific measured time series, and outputs if (and where) there is an *anomaly*, a rare event where the time series somehow "breaks" its normal behavior, due to some external influence which can be modeled with noise or some rare, hence important and interesting, event.

More specifically, there are three types of phenomena, shown in Figures 1, 2 and 3, that we want to detect. The three examples depicted show how anomaly can be seen as an "unexpected behavior" of the series of measurement, which basically means: extra information, or entropy, in the signal.

Autoencoders

An autoencoder algorithm, as depicted in Figure 4, consists of two components: an *encoder* that projects the input data into a lower-dimensional (compressed) latent representation, and a *decoder* that reconstructs an output of the same dimension as the input. Ideally, the latent representation captures all the relevant information, enabling the output to closely resemble the input according to a chosen similarity metric. To achieve this, the autoencoder must be trained on a rich dataset that shares the same characteristics as the target data—in this case, a dataset of time series with similar properties, measurements in time of the same physical phenomenon.

Autoencoders are highly effective for anomaly detection. When presented with input data containing anomalies (extra information), the autoencoder struggles to encode it efficiently. This inefficiency results in reconstructed output data that is less similar to the input compared to non-anomalous data.

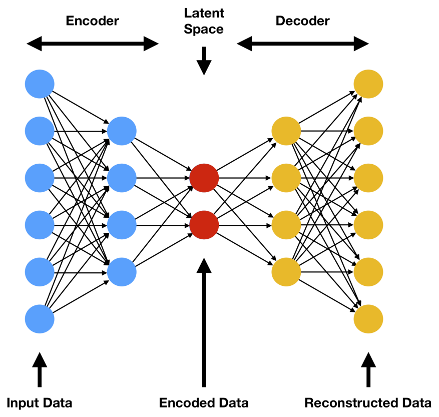
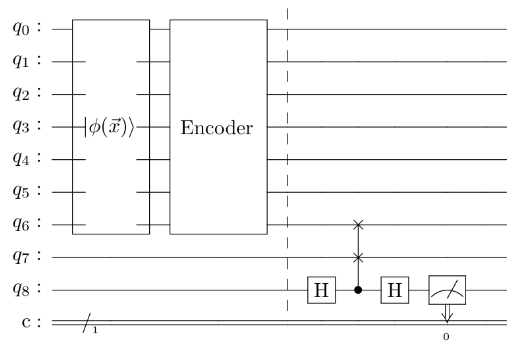


Figure 4: The structure of a classical autoencoder.

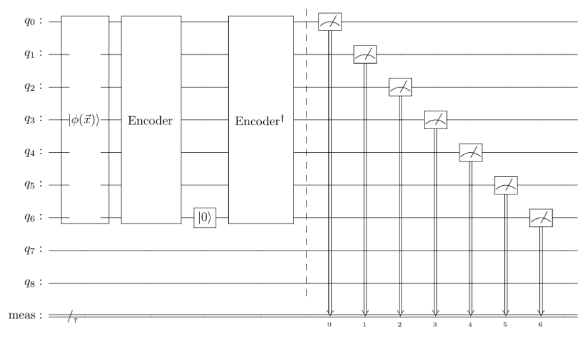
Quantum Autoencoders

The article proposes a quantum-enhanced version of the autoencoder algorithm. **The classical deep learning version** consists in many layers of perceptrons (the 2 layers in Figure 4 are just a semplification) which means that a **very high number of parameters has to be trained**. As already pointed out in the introduction, the main reason to introduce this quantum version, is a dramatic reduction of utilized parameters, 60-230 times fewer.

Below a representation of a quantum autoencoder architecture, and in the following sections a detailed description of its components and their purpose, referring specifically to the time series analysis application.



(a) Trainable quantum autoencoder architecture



(b) Trained quantum autoencoder architecture

Figure 5: The figures depict a 7-qubit quantum autoencoder circuit. Despite requiring 9 qubits for training, it is termed a 7-qubit autoencoder due to the number of qubits in the state preparation and encoder sub circuit. Figure 5a illustrates the circuit used during training, while Figure 5b shows the final autoencoder capable of reconstructing input states. Encoder parameters optimized during training the archictecure in Figure 5a are transferred to the setup in 5b. In both figures, $\phi(\vec{x})$ represents the state preparation procedure, and *Encoder* denotes a quantum variational sub-circuit with consistent architecture in both setups. Additionally, $|0\rangle$ signifies qubit reset (qubit 6), and *Encoder*† denotes the conjugate transpose of the encoder, serving as the decoding component.

(Classical) preprocessing

This step is executed on classical computing architectures (CPU) and is NOT represented in Figure 5.

First, we generate sliding windows $\vec{w_i} = [w_i, w_{i+1}, ..., w_{i+2^K-1}]$ of fixed size 2^K spanning from index i to $i + 2^K - 1$ of the time series, where $K = 7$ qubits incorporated in the quantum autoencoder. By sliding the window along the time series with a certain step size (which in practice consists in choosing specific values for the variable i to attain), overlapping segments are created, enabling a more comprehensive understanding of the data. The window width and step size can be adjusted to suit the specific characteristics of the data being studied. **In the following experiments we preprocess the data into sliding windows of size 128 with a stride of 1, with each window serving as input to the autoencoder.**

We refrain from normalizing the time series data, since this operation is subsequently performed during the quantum state preparation process, as it would be appropriate to do when using a classical deep learning-based autoencoder.

Quantum state preparation

This and the next following steps are performed on quantum hardware platforms (QPU) and are shown in Figure 5 represented as quantum gates.

In this step we transform classical input data into the initial quantum state: each window $\vec{w_i}$ is transformed into a quantum state $|\phi_i\rangle$ via **amplitude encoding**, efficient in encoding a large number of input features with a minimal qubit requirement. **In this specific instance, the feature space resides in the $2^K = 2^7 = 128$-dimensional Hilbert space $\mathcal{H}^{128}$**. This method requires only $\log_2(N)$ qubits to encode N datapoints and establishes a direct correspondence between the input and the probability amplitudes of the K-qubit state. Specifically, given a window $\vec{w_i} = [w_i, w_{i+1}, ..., w_{i+2^K-1}]$, the prepared quantum state using amplitude encoding is expressed as

$$|\phi_i\rangle = \sum_{j=0}^{2^K-1} \frac{w_{i+j}}{\langle w_i|w_i\rangle} |j\rangle \tag{1}$$

. Here i indicates which window we are encoding and $|j\rangle$ represents the collective state of the $K = 7$ qubits, in base 10 (e.g. $|0101011\rangle = |j = 43\rangle$).

It is important to highlight that the state preparation step does not involve trainable parameters for autoencoding and is entirely deterministic in nature.

Quantum encoding

As in the general classical case, the *Encoder* block compresses the data to a lower dimension. In the quantum case (Figure 5) the encoder gate is applied to the initial quantum state and is built upon multiple variational quantum gates: it is **trained to map from a 2^7-dimensional Hilbert-Space to 2^6**. Ideally, after application of the encoder qubit 6, called *trash state*, does not hold any information (i.e. it is equal to the zero state) resulting in all of the information condensed into qubits 0 to 5, which represent the *latent space.*

After the Encoder block, the trash state(s) is manually reset to the $|0\rangle$ state, as shown in Figure 5b. In this study the authors conservatively chose an encoder architecture with just one trash state, but one might use smaller latent spaces, so more trash states, for less entropic time series.

Training

Resetting of the trash qubit(s) during the encoding process leads to the loss of all information stored in the corresponding qubits and influences the decoding process. Therefore, to mitigate such information loss, an ideal scenario entails the encoder to efficiently map all relevant information to the qubits associated with the latent space before the resetting operation occurs. The mapping is accomplished using the circuit depicted in Figure 5a, which incorporates the state preparation block and the encoder circuit of the desired autoencoder. This circuit is extended by introducing a reference qubit (qubit 7) initialized as $|0\rangle$ and an ancillary qubit (qubit 8) for executing the **Swap-Test** , which determines whether a pair of states are inequivalent: **upon measuring qubit 8, if qubits 6 and 7 are equal, the probability of obtaining a measurement outcome of 0 for qubit 8 is 1, whereas if qubits 6 and 7 are orthogonal, the probability of observing a 0 for qubit 8 is 1/2**. Training the encoder to effectively map all information onto the qubits representing the latent space, requires maximizing the number of zeros observed in qubit 8, indicating that qubit 6 closely aligns with the initialized $|0\rangle$ state of qubit 7.

Quantum decoding

The unitary property of the encoding transformation removes the necessity of training a decoder, in contrast to classical deep learning-based autoencoders. The decoder, in this context, becomes a straightforward inversion of the transformation accomplished by applying the compressed data to the conjugate transpose of the encoder. The obtained quantum state in the ideal case is equal to the quantum state after applying the state preparation procedure.

(Classical) postprocessing and classification

This step is, like the Preprocessing, executed on classical computing architectures (CPU) and is NOT shown in Figure 5.

After state preparation, the time windows are autoencoded using either the trainable or trained architecture shown in Figure 5a and 5b, respectively. Utilizing the trained architecture, which involves the inverse transform after resetting the trash qubits, resembles classical deep learning counterparts, allowing for the reconstruction of the input and subsequent classification based on the reconstructed time windows. **For anomaly detection without input reconstruction, the trainable setup using only the Swap-Test can be employed. Here, the number of measured 1s should be minimal for benign data observed during training, in contrast to unseen anomalies**.

Upon autoencoding the initial quantum state $|\phi_i\rangle$ and deriving the output state $|\phi'_i\rangle$, we either compute the mean squared error

$$\epsilon_i = \frac{\langle \phi_i - \phi'_i | \phi_i - \phi'_i \rangle}{2^K} \tag{2}$$

between input and reconstructed output in case we use the autoencoder architecture depicted in Figure 5b. If we use the Swap-Test measurements we generate a vector

$\vec{\epsilon} = [\epsilon_1, ..., \epsilon_i]$ of the same length as the number of time windows we processed, where ϵ_i is the Swap-Test measurement for the particular time window starting at $t = i$.

Commonly, a threshold-based classification approach is used, wherein a time step i is designated as anomalous if its corresponding post-processed reconstruction error p_i surpasses or equals a pre-defined threshold t.

Using a window-based methodology, an anomaly is considered a valid detection of the anomaly if it starts within the valid detection range.

Experimental results

While in the study [1] extensive quantum simulations and real quantum hardware experiments have been conducted, the follow experiments will be only simulated, with the main objective of seeing the theoretical description contained in the previous chapters put to practice using real world datasets.

Classical autoencoder

To enable a comparative analysis between quantum autoencoders and classical deep learning based autoencoders, we establish a classical baseline model, **with 4,686 parameters**, shown in Figure 6, implemented using the PyTorch framework.

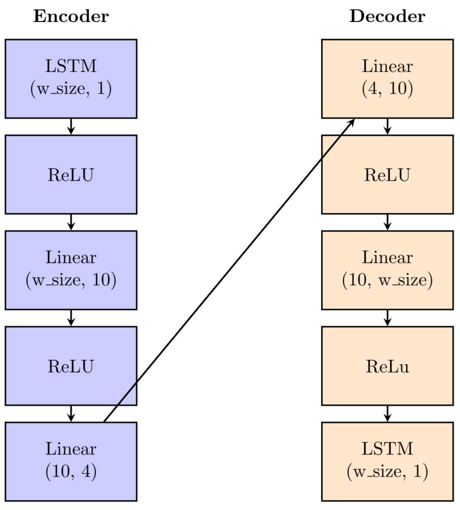


Figure 6: Illustration of the classical deep learning autoencoder used as the baseline in this study. The window size is set to 128.

Training the classical autoencoder on the “FISH” train dataset in [2], it takes 50 epochs to reach a loss (mean square error between input and output) of 0.0107. The trained classical autoencoder performs well on the unseen test data, as shown in Figure 7, which represents the worst-reconstructed (greater MSE) non-anomalous time series, and successfully finds the one and only anomaly in the test data, Figure 8. This anomaly is easily spotted as it corrisponds to a huge spike in the MSE, represented in Figure 9, which means that, as anticipated in the previous chapters, the anomalous time series is reconstructed much more poorly than the non-anomalous ones by a classical autoencoder trained on non-anomalous data.

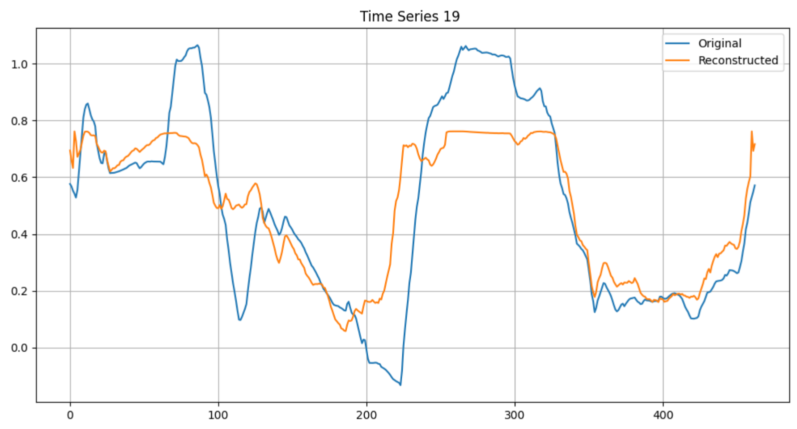


Figure 7: The non-anomalous time series from the "FISH" test dataset in [2], reconstructed via the classical autoencoder, which has the greater mean square error.

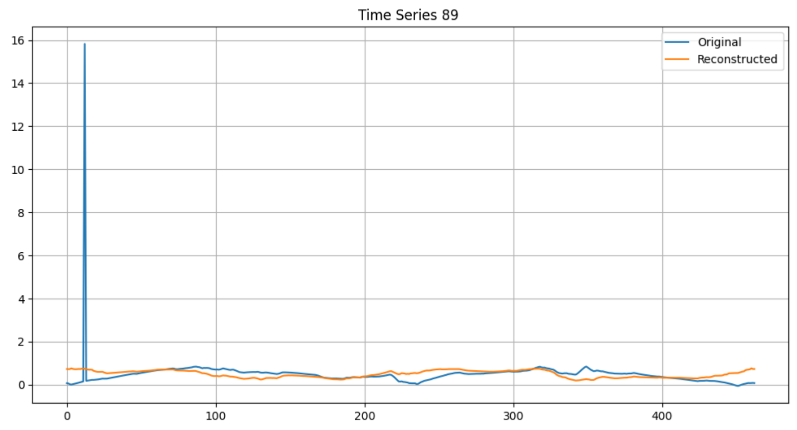


Figure 8: The anomalous time series from the "FISH" test dataset in [2], reconstructed via the classical autoencoder.

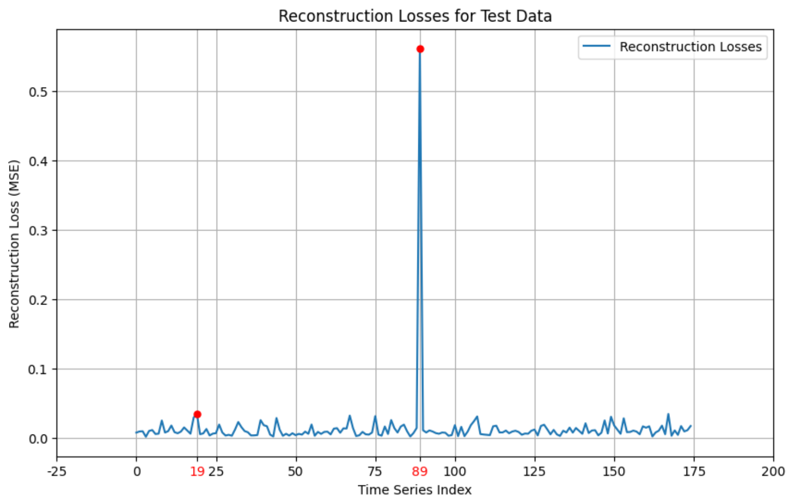


Figure 9: The Mean Square Error between the Original and the Classical Reconstructed Time Series from the "FISH" test dataset in [2].

Quantum autoencoder

For the quantum autoencoder, it used only the higher performing ansatz from article [1], which is shown in Figure 10, substituting 1) with Amplitude Embedding Pennylane function, since the the experiments were was not performed on real quantum hardware. **The variational block contains 21 trainable parameters.**

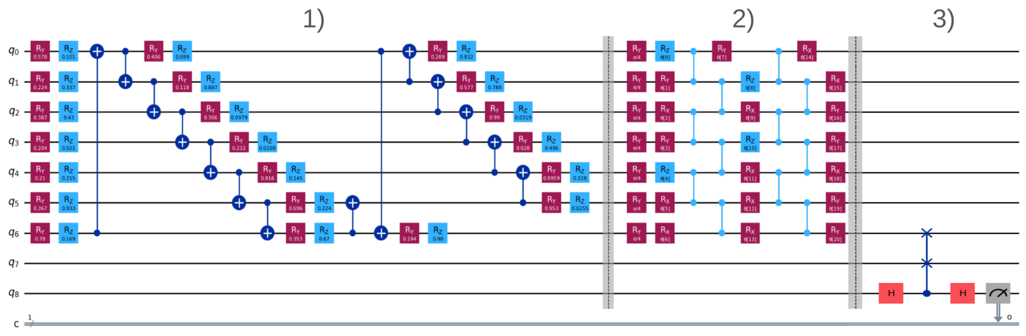


Figure 10: Illustration of the quantum autoencoder used in the article [1]. Component 1) represents the quasi-encoded time window using the EfficientSU2 circuit layout. The parameter values in this section depend on the input data and remain constant during training. Component 2) illustrates the trainable encoder of the quantum autoencoder, which, in this case, is based on the PauliTwoDesign with two repetitions. These parameters are adjustable and are optimized during training on the actual hardware. Lastly, component 3) corresponds to the Swap-Test which measures how well the encoder circuit encodes the quasi-encoded input data. Ideally the likelihood of measuring qubit 8 in state 1 is close to 0.

Training this quantum autoencoder algorithm takes a few hours per epoch, due to the complexity of the simulation of 9 qubits systems, but the loss (in this case it is the probability to measure the state $|1\rangle$ in the qubit 8 of the trainable architecture in Figure 5a) decreases very fast to values around 10^{-7}. For this reason early stopping with a loss threshold of 0.001 was used: this makes the quantum autoencoder training almost 10 times faster than the classical case, even choosing a threshold loss which is almost 10 times lower.

As previously pointed out, the trained encoder can be used on its own to detect anomalies: they corrispond to higher probabilities to measure the state $|1\rangle$ in the qubit 8 of the trainable architecture in Figure 5a. As shown in Figure 11 this method successfully detects the anomalous time windows, they correspond to the huge spike.

On the other hand, one might want a more expressive output, for example to enable comparison with the classical autoencoder performances: in this case it is possible to pass the original time windows through the entire autoencoder, Figure 5b, and obtain the reconstructed time windows reported in Figure 12 and 13. Again, as in the classical case, it can be calculated the mean square error between original and reconstructed time windows, plotted in Figure 14. The reconstructions and the MSE can be used, as in the classical case, to spot the anomaly, but they look more noisy and do not really represent an enhancement to the classical autoencoder in terms of accuracy. The best usage of the quantum autoencoder architecture seems to be the simple swap test probability measurement, which benefits from both the training speed and from a high accuracy.

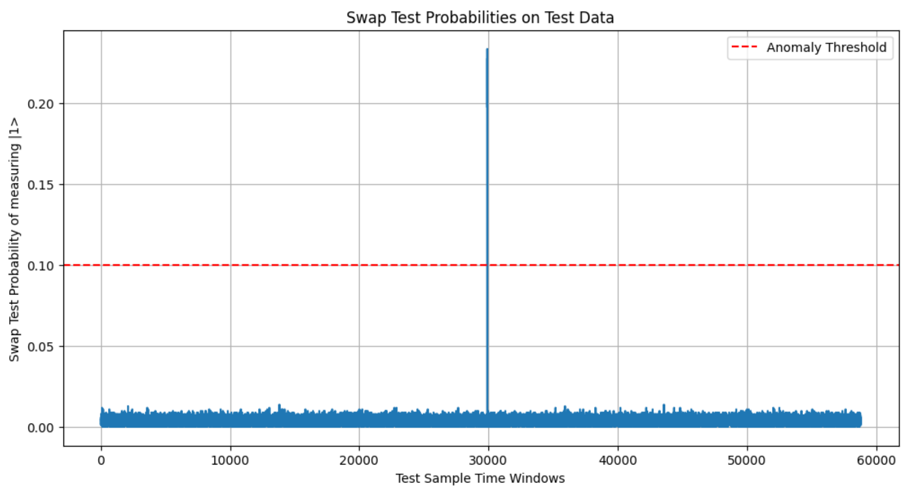


Figure 11: Anomaly detection using the probability of measuring the qubit 8 in the state $|1\rangle$, for the "FISH" test dataset.

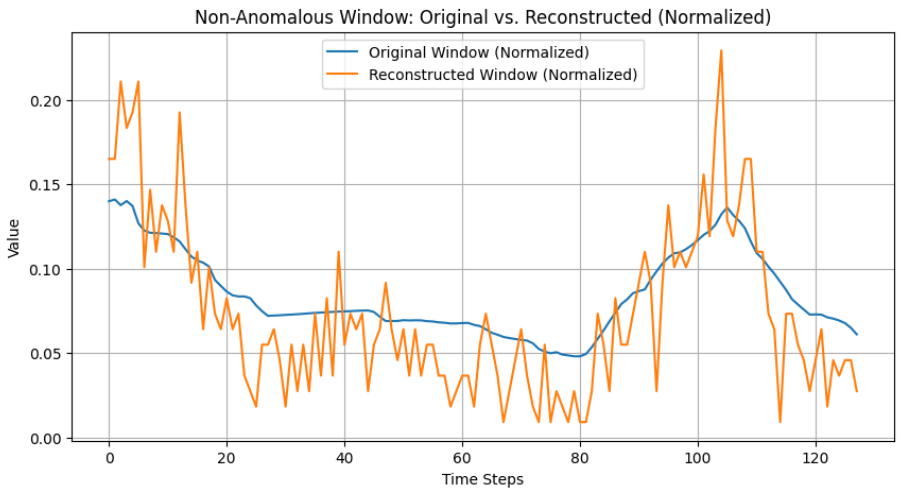


Figure 12: A non-anomalous time window belonging to a time series from the "FISH" test dataset in [2], reconstructed via the quantum autoencoder.

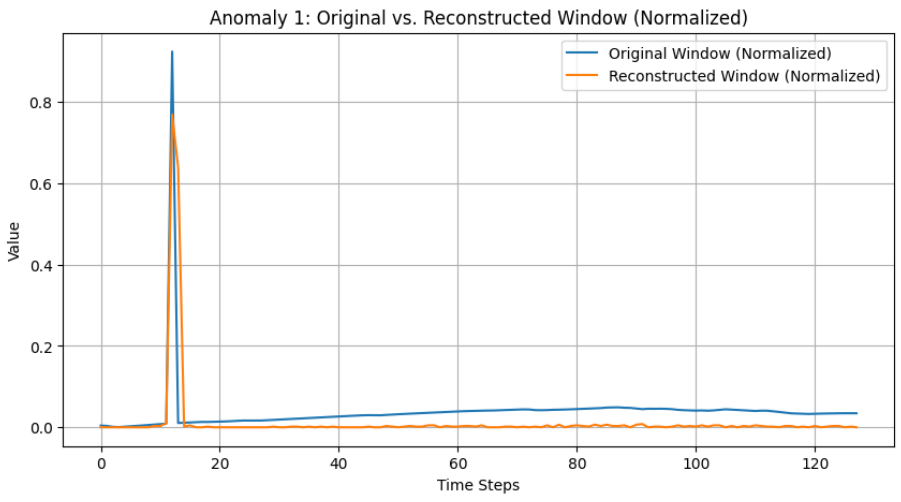


Figure 13: One of the time windows belonging to the anomalous time series from the "FISH" test dataset in [2], reconstructed via the quantum autoencoder.

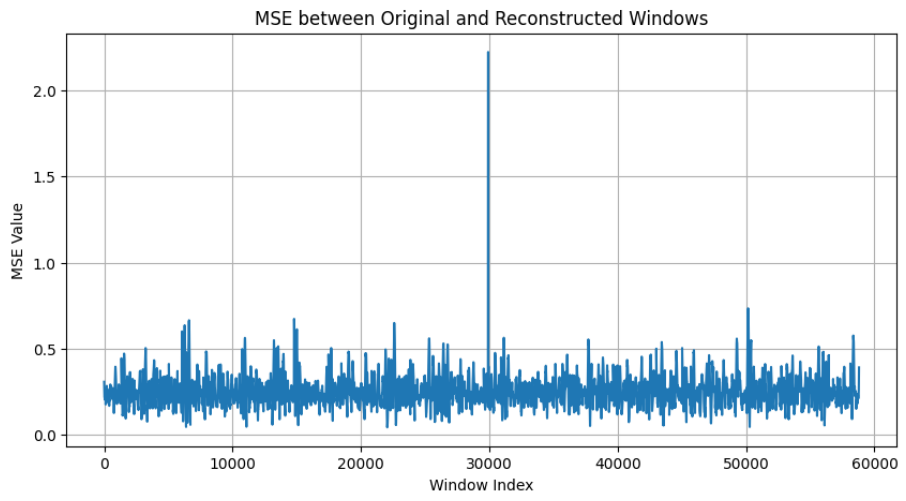


Figure 14: The Mean Square Error between the Original and the Quantum Reconstructed Time Series from the “FISH” test dataset in [2].

Further experimental results

For a more complete comparison between the classical and quantum autoencoder structures proposed, more experiments have been conducted, making use of another dataset, again taken from the archive [2]. As evident from Figure 8, the type of anomaly present in the “FISH” test dataset is a big spike, a.k.a. point anomaly (Figure 1). This type of anomaly is the easiest to spot, in fact its detection would not even require a machine learning setup, but just a little standard practice data treatment.

Here are reported the performances of the two classical and quantum autoencoders, making use of the dataset “Adiac”, which test set contains, as shown in Figure 16 a much more elusive collective anomaly (Figure 3).

Classical autoencoder

Training the classical autoencoder on the “Adiac” train dataset in [2], it takes 250 epochs to reach a loss (mean square error between input and output) of 0.0013. The trained classical autoencoder performs well on the unseen test data, as shown in Figure 15, and successfully finds the one and only anomaly in the test data, Figure 16. This anomaly corrisponds to the spike in the MSE, represented in Figure 17. As stated before, this kind of anomaly is not easily detected without making use of an autoencoder algorithm.

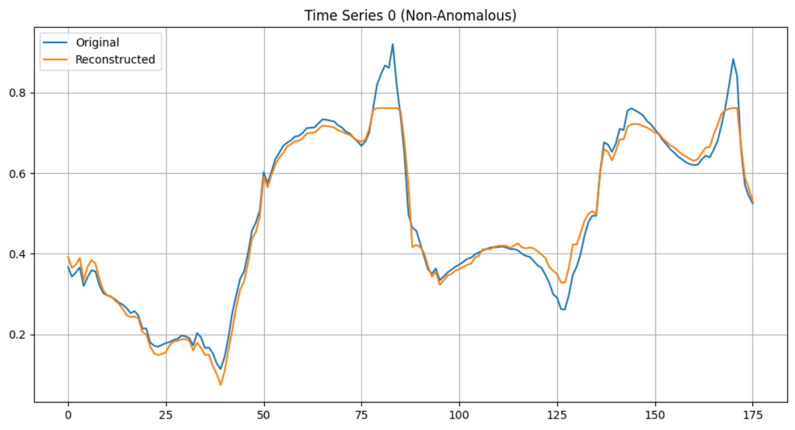


Figure 15: A non-anomalous time series from the “Adiac” test dataset in [2], reconstructed via the classical autoencoder.

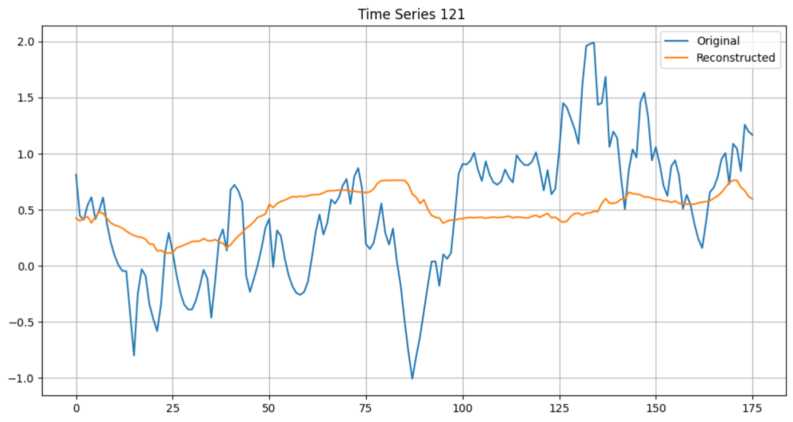


Figure 16: The anomalous time series from the “Adiac” test dataset in [2], reconstructed via the classical autoencoder.

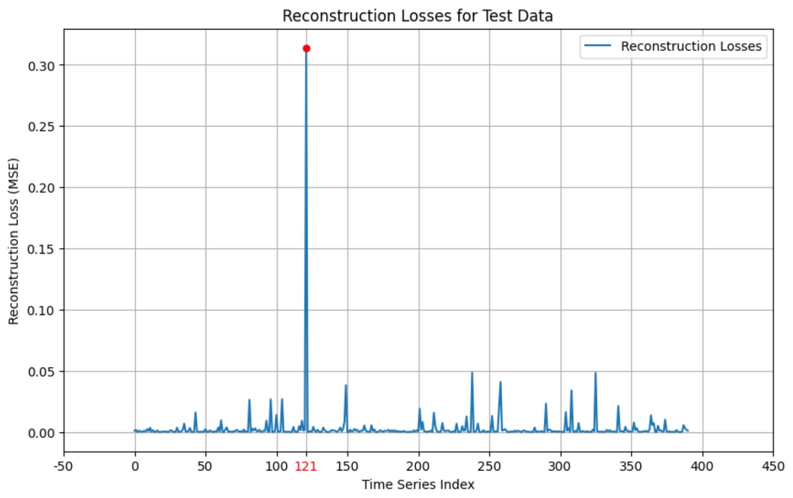


Figure 17: The Mean Square Error between the Original and the Classical Reconstructed Time Series from the “Adiac” test dataset in [2].

Quantum autoencoder

For the quantum autoencoder, once more only the higher performing ansatz from article [1], which is shown in Figure 10, was used, substituting 1) with Amplitude Embedding Pennylane function, since the the experiments were was not performed on real quantum hardware. Again a loss threshold of 0.001 was set for early stopping.

As shown in Figure 18 or, less clearly, in Figure 21, this quantum autoencoder ansatz successfully detects the anomalous time windows. Hereby are reported also the reconstructed time windows in Figure 19 and 20.

The reconstructions and the MSE, as in the “FISH” dataset previous experiment, look quite noisy and do not really represent an enhancement to the classical autoencoder in terms of accuracy. The best usage of the quantum autoencoder architecture seems to be, once again, the simple swap test probability measurement, which benefits from both the training speed and from a high accuracy.

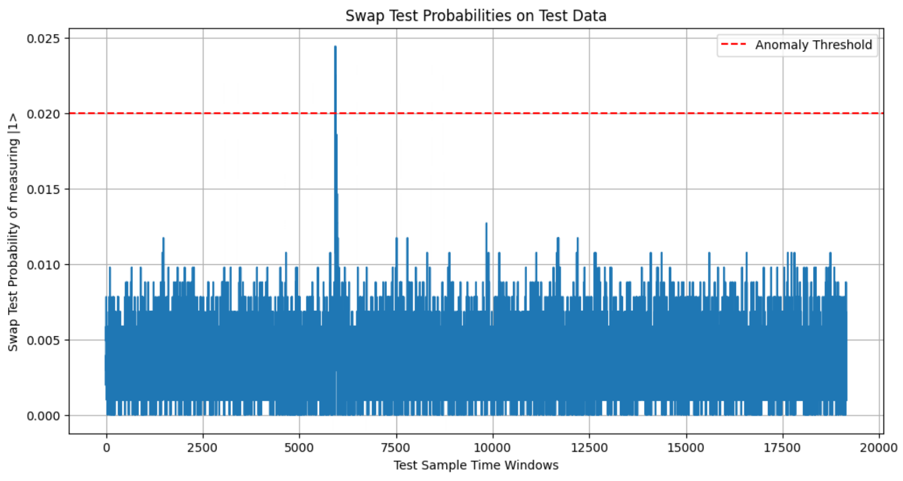


Figure 18: Anomaly detection using the probability of measuring the qubit 8 in the state $|1\rangle$, for the "Adiac" test dataset.

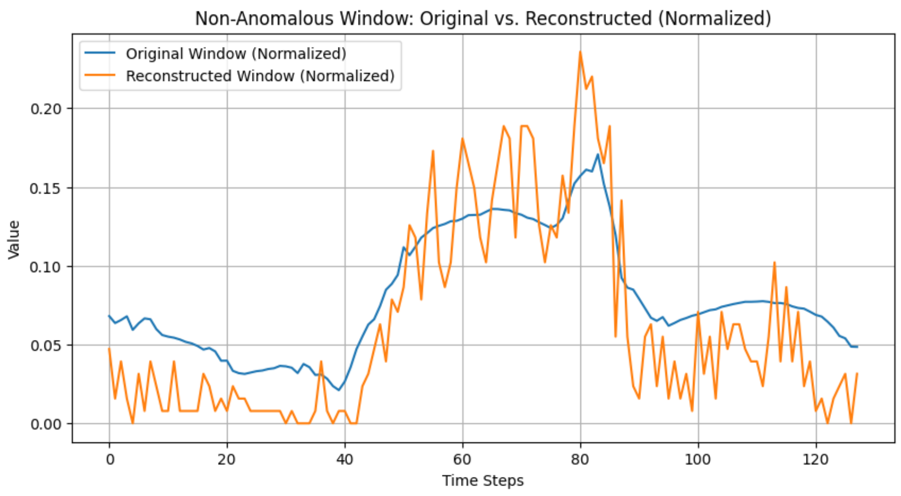


Figure 19: A non-anomalous time window belonging to a time series from the "Adiac" test dataset in [2], reconstructed via the quantum autoencoder.

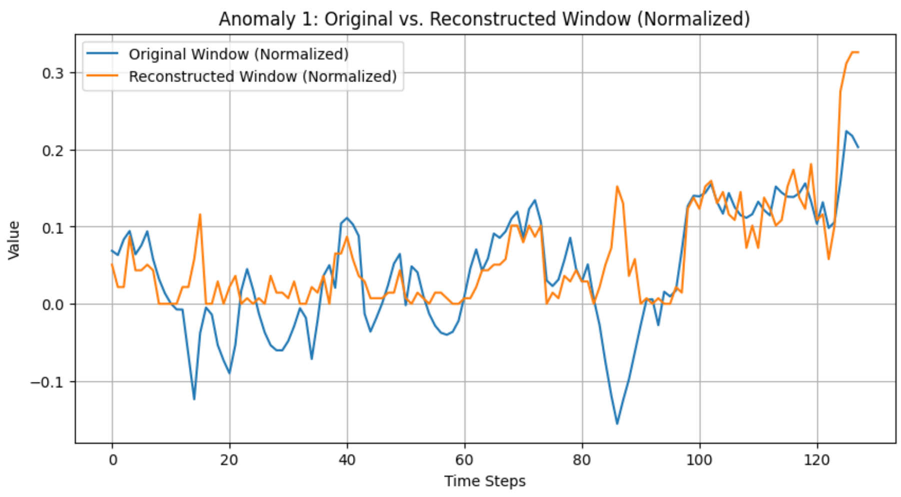


Figure 20: One of the time windows belonging to the anomalous time series from the "Adiac" test dataset in [2], reconstructed via the quantum autoencoder.

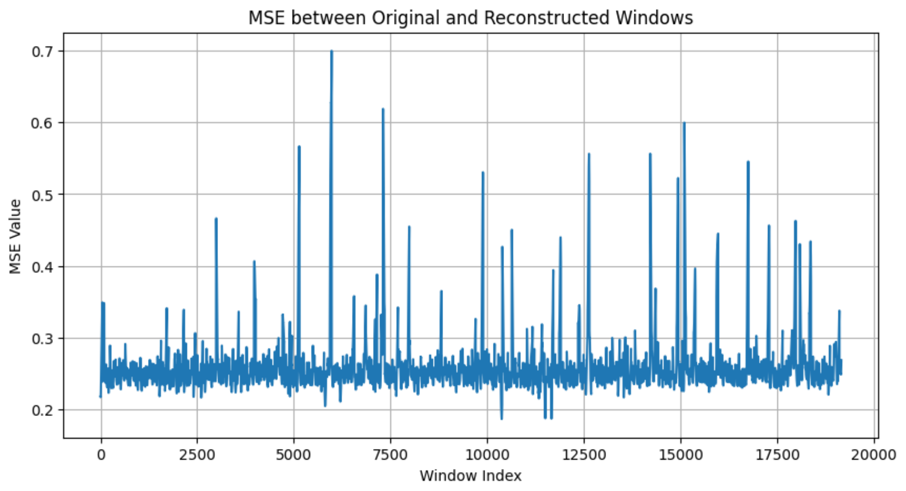


Figure 21: The Mean Square Error between the Original and the Quantum Reconstructed Time Series from the “Adiac” test dataset in [2].

Conclusion

This report shows a step-by-step detailed reconstruction of the classical and an ansatz of quantum autoencoder proposed by Robin Frehner and Kurt Stockinger in the article "Applying Quantum Autoencoders for Time Series Anomaly Detection" [1]. The two algoritms were trained and tested on datasets not reportedly studied by the authors, but from the same UCR time series classification archive [2], confirming their results.

More specifically, experiments were conducted on the "FISH" and on the "Adiac" dataset, both containing no anomalies in the training data and one and only one in the test data. The first dataset contains one *point anomaly*, easier to spot, while the second contains a more subtle *collective anomaly.* The classical (4,686 parameters, ≥ 50 epochs of training) and the quantum (21 parameters, ≤ 1 epoch due to early stopping) autoencoders successfully detected both the anomalies.

Bibliography

[1] Robin Frehner and Kurt Stockinger. Applying quantum autoencoders for time series anomaly detection, 2024.

[2] Hoang Anh Dau, Eamonn Keogh, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, Yanping, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, Gustavo Batista, and Hexagon-ML. The ucr time series classification archive, October 2018. `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/`.