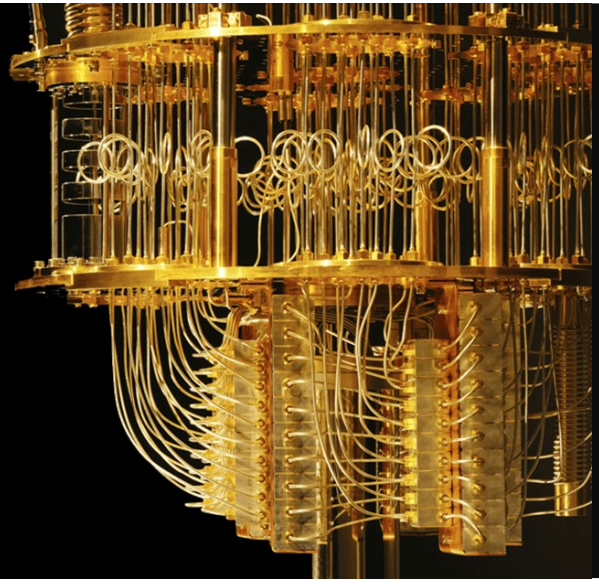
The aims of this session:

To identify design decisions taken in the process of building a complex quantum machine learning model



Introduction to QuTSAE
QuTSAE design choices
Architectural choices
Input encoding choices
Output / cost function choices
Encoder / decoder ansatze choices
Optimization / training choices
Summary of results
Conclusions and future work

Design Considerations for Denoising Quantum Time Series Autoencoder (QuTSAE)

Jacob L. Cybulski
School of IT, SEBE, Deakin University, Melbourne, Australia

Sebastian Zając
SGH Warsaw School of Economics, Warsaw, Poland

2-4 July 2024, Málaga Spain

Quantum Time Series Autoencoder

QuTSAE: wordplay on the Hopi notion of *qatsi = life*

Autoencoders (AE) are ML models able to to compress input into its essential features and then recover the original info

They lose the infrequent or insignificant info, such as anomalies and noise

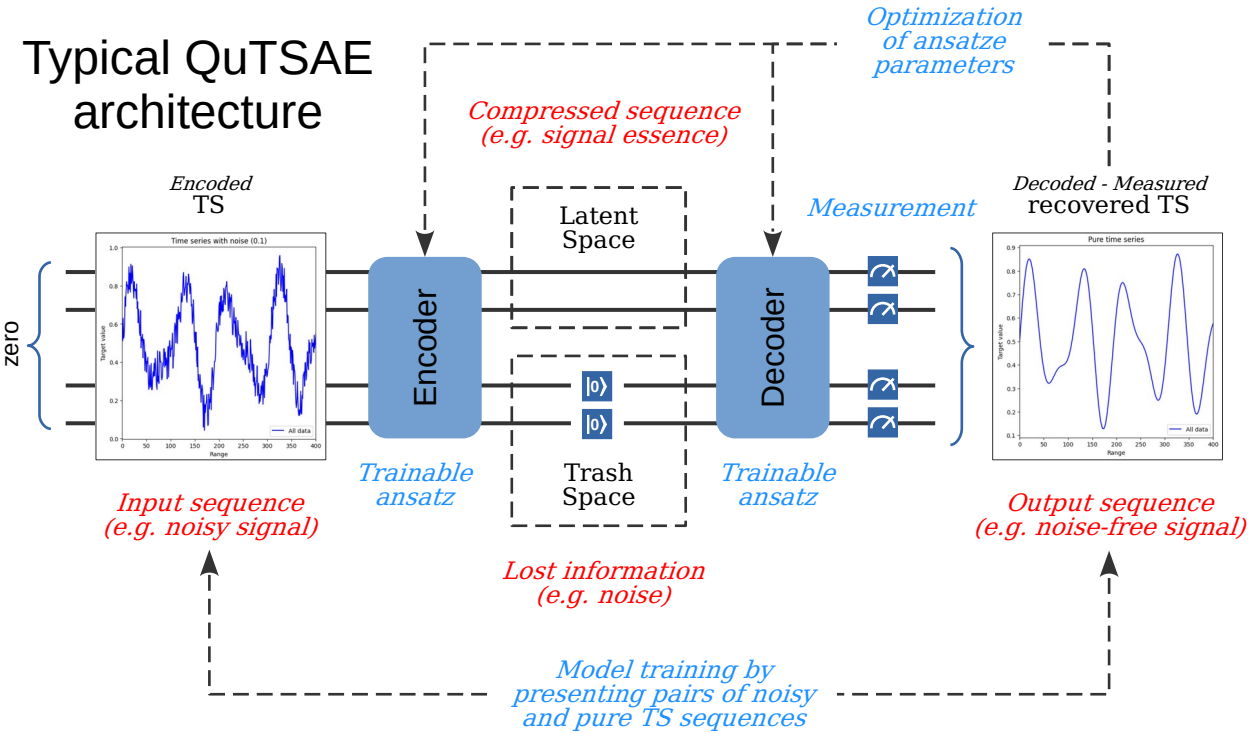
Used for data denoising and anomaly detection, e.g. in images and signals

There are few applications of QML methods to time-series analysis, QAE even fewer

QuTSAEs have the potential to deal with complex noise and anomaly patterns

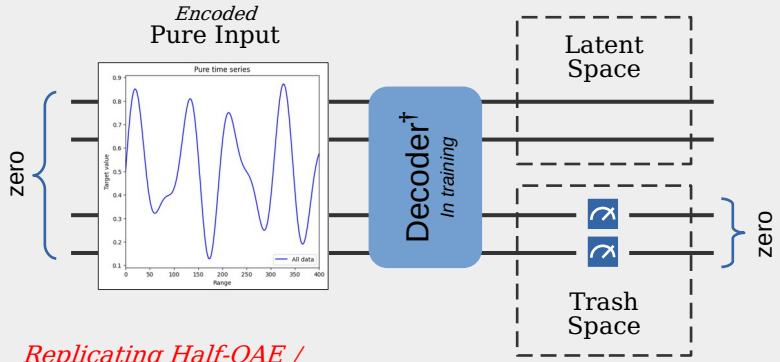
Training of QuTSAEs is difficult, due to:

- Many features
  (lots of qubits and/or parameters)
- Complex measurement strategies
- Unsupervised learning
  (we do not know what is noise)
- Possibility of barren plateaus



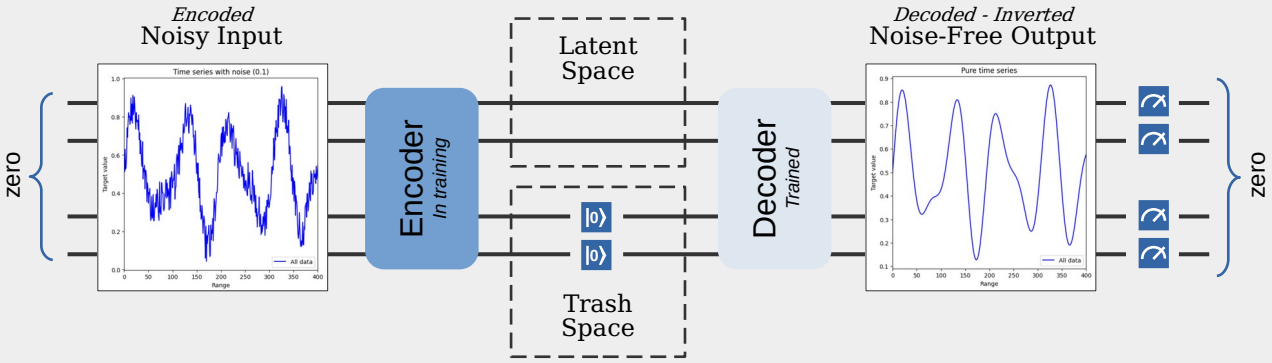
In QuTSAE development, the key concerns include:
overall model architecture, data encoding and decoding, ansatz design and its parameters optimisation strategy

Alternative Architectures

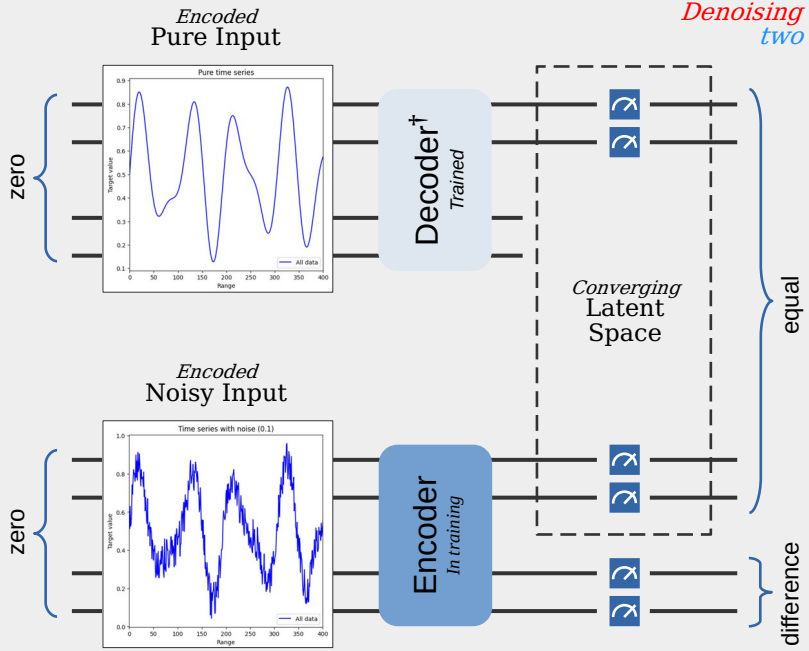


Replicating Half-QAE /
single stage (pure data only)

Approximating or Denoising Stacked half-QAEs /
two stages (pure+noisy data)



Denoising with half-QAE Sidekick /
two stages (pure+noisy data)



A novel approach to QuTSAE training (exclusive preview)

Merriam-Webster Dictionary (acc 3 March, 2024):

Someone's sidekick is a person closely associated with another as a subordinate or partner

We can train a pure QAE by training its half by converging trash info to zero, the other half is its inverse.

We can train a noisy half-QAE by stacking it with a pure half-QAE

We can also side-train a noisy half-QAE by converging its latent space to a pretrained pure half-QAE

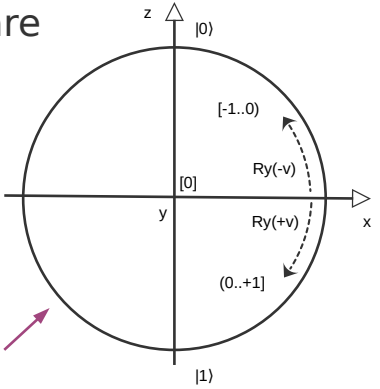
Input encoding and output decoding for QuTSAE processing

Note similarities between input encoding and output decoding

In general, QuTSAE input and output is a collection of real values (floats)

There are many different quantum data encoding / state preparation methods, e.g.

- *basis encoding*, with qubits acting as bits in the encoded number (logical / int) to be processed later in the circuit
- *amplitude encoding*, where each data point is encoded as expectation value of multi-qubit measurement (int / float)
- *QRAM encoding*, where possible inputs are pre-coded in a circuit, and used by ref
- *angle encoding*, where qubit rotation (float) represents the value of data
- Our experiments used angle encoding relative to $|+\rangle$ state with values ∈ [-1, 1] coded as rotations up (<0) or down (≥0)



Output decoding relies on the interpretation of the model measurement, which is linked to the cost function and its use.

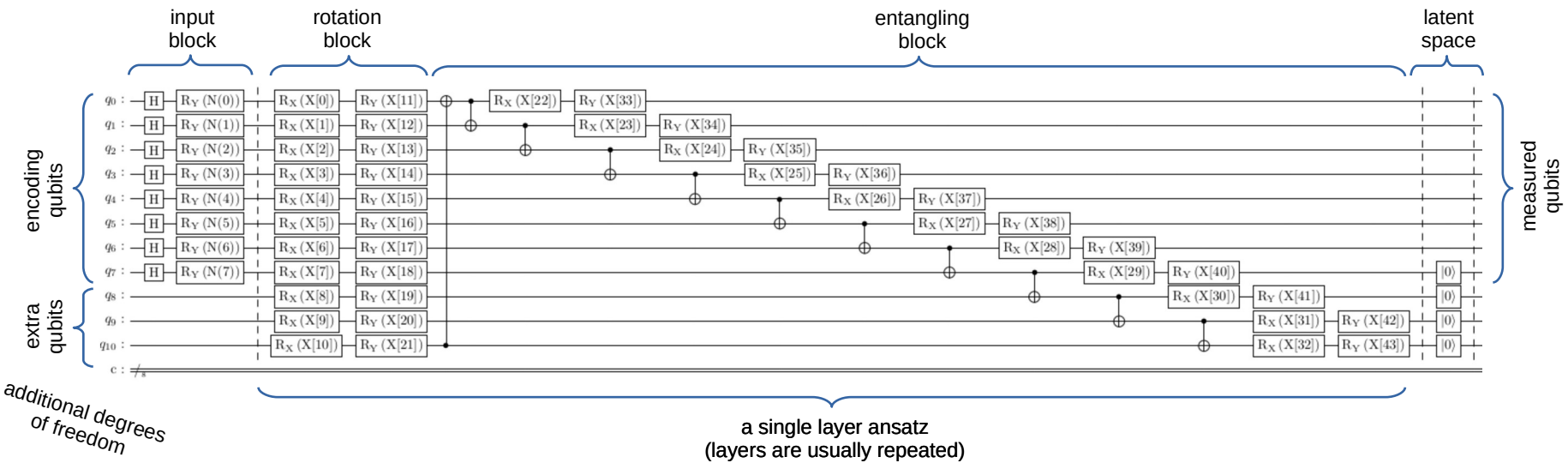
In QuTSAE to determine the angular state of individual qubits for model training and testing, we can use:

- *Swap test* to compare the state of the selected qubits to the initial (zero) state of unused qubits
- *Measurement and calculation*, which could rely on the probability (count) distribution of expectation values, followed by calculation of conditional probabilities of each qubit's state, from which it is possible to derive the angular state of each qubit.

Maria Schuld and Francesco Petruccione. *Machine Learning with Quantum Computers. 2nd ed*. Springer, 2021. http://link.springer.com/book/10.1007/978-3-030-83098-4.

Anatomy of a QuTSAE Ansatz

QAE encoder and decoder



QuTSAE encoder and decoder are often symmetric (although this is not necessary)

They are parametrized circuits (ansatze), arranged into layers of trainable rotation blocks and entangling blocks

Ansatze may be of a different size than the requirements of input/output blocks

The selection of the optimizer of ansatze parameters requires some preliminary investigation of their effectiveness

This depends on the model architecture, ansatz design, data encoding and decoding, as well as the nature of training data

In our project we evaluated gradient based optimizers (ADAM and SPSA) as well as linear and non-linear approximation methods (such as COBYLA and BFGS) – COBYLA was adopted

Experiments
with Denoising QuTSAE

Subsequently, a series of over 60 (Qiskit) experiments were conducted to determine the optimum QuTSAE model characteristics

First, it was required to determine the time-series window size, which implied the size of QuTSAE model input and output blocks

Then three circuit parameters were varied, i.e. the size of latent (and trash) space, the number of additional qubits required by ansatze, and the type and number of rotational (y and xy) / entangling layers

The optimum model parameters were selected based on the model validation scores (MAE)

The best QuTSAE model's performance was comparable to (but not better than) the best equivalent classical model (14 additional experiments with PyTorch models)

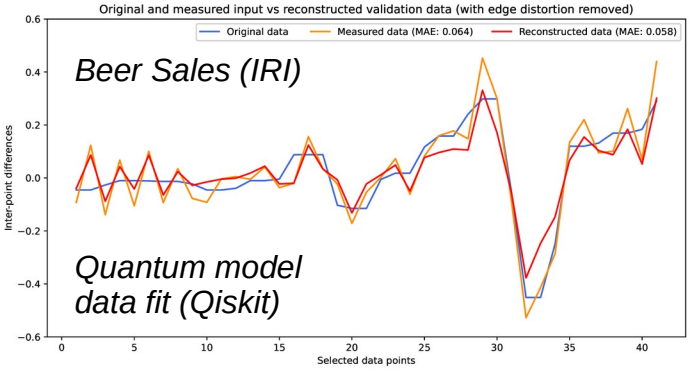
Qiskit state vector simulator
Best quantum model (7, 3, 2)
Number of parameters: 180
Number of iterations: 2,000
Speed of model training: 15 mins

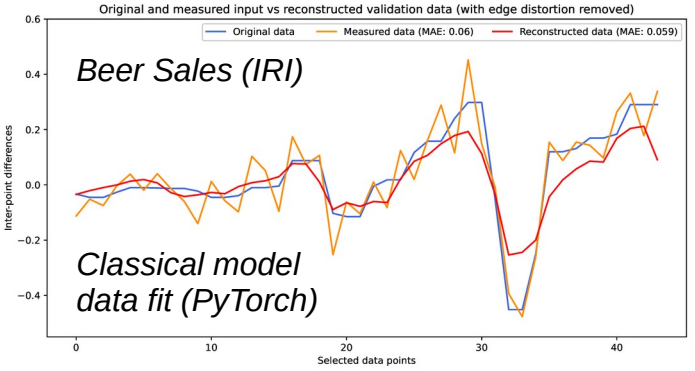
PyTorch
Equivalent classical model (7)
Number of parameters: 9,741
Number of iterations: 30,000
Speed of model training: 20 secs

Table 1: Cost and scoring results (top 4 models in Ex. 1, 2a and 2b experiments)

Experiment			Min	Training				Validation			
Lat	Aw	Reps	Cost	R2	RMSE	MAE	MAPE	R2	RMSE	MAE	MAPE
Ex. 1 (Qiskit/QNN)											
7	3	2	0.089	0.767	0.081	0.059	3.592	0.803	0.074	0.058	1.421
7	3	1	0.054	0.742	0.085	0.067	4.471	0.782	0.078	0.064	1.946
7	5	1	0.078	0.738	0.086	0.071	3.416	0.767	0.081	0.068	2.455
7	4	1	0.065	0.658	0.097	0.074	3.880	0.723	0.088	0.072	2.191
Ex. 2a (Qiskit/QNN)											
7	3	1	0.054	0.742	0.085	0.067	4.471	0.782	0.078	0.064	1.946
6	3	1	0.070	0.399	0.130	0.087	2.745	0.442	0.116	0.088	2.367
4	3	1	0.063	0.428	0.126	0.085	3.346	0.331	0.137	0.094	1.568
5	3	1	0.066	0.455	0.123	0.095	4.603	0.403	0.129	0.097	2.654
Ex. 2b (Qiskit/QNN)											
7	3	2	0.089	0.767	0.081	0.059	3.592	0.803	0.074	0.058	1.421
8	3	2	0.086	0.761	0.083	0.066	4.743	0.741	0.085	0.068	1.952
2	3	2	0.086	0.752	0.085	0.068	4.668	0.690	0.090	0.072	2.490
5	3	2	0.105	0.396	0.132	0.088	3.122	0.461	0.115	0.079	1.473
Ex. 3 (PyTorch/MLP, lat=7)											
Enc=3/Dec=3			0.012	0.996	0.010	0.006	0.225	0.751	0.081	0.059	1.310

Summary





Research insights

- We have discussed design decisions taken in the development of denoising quantum time series autoencoders
- Input encoding strategy needs to consider the output decoding strategy, i.e. measurements and their interpretation
- Methods of measuring and interpreting model's quantum state are essential for training and testing the model
- Ansatz architectural properties must fit the model function
- Ansatz circuit width, depth, the number of trainable parameters, as well as the required degree of freedom (extra qubits) determine the model performance
- All QuTSAE design choices impact its performance in training and validation
- QuTSAE models are not better but merely approach the performance of classical models
- The majority of insights derived from this study are applicable to the design of many other QML models

Future work

- Explore QML platforms that provide better support for VQA and models optimisation
- Introduce tighter integration between classical and quantum ML methods, which seem to result in better performing VQA models

Preview of the current work

Improved optimisation of the model parameter space

PennyLane/ PyTorch
approach to QuTSAE development

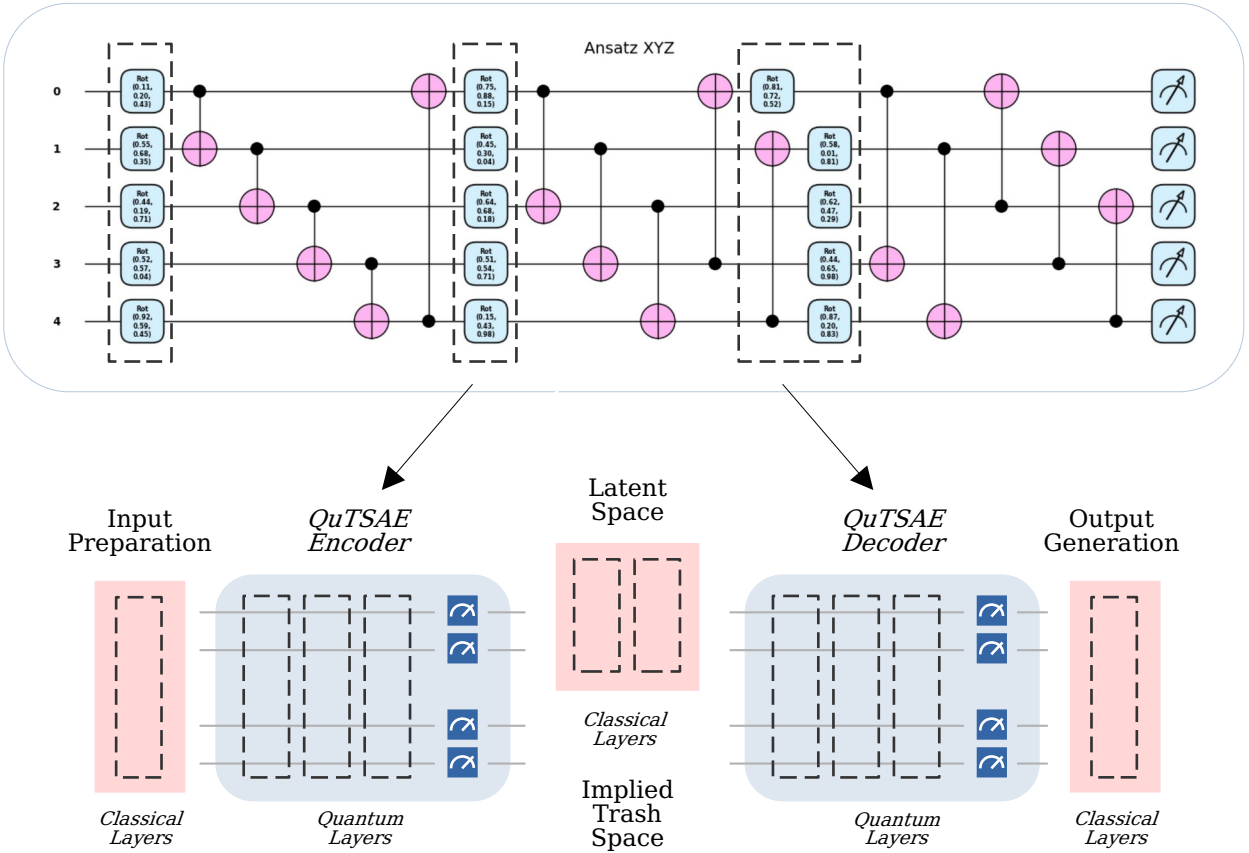
An approach adopted in the QuTSAE creation was to rely on the VQA development in Qiskit

One of the issues found to affect the QuTSAE training performance is:

- *The lack of quantum model transparency which hampers the performance of classical optimizers*

The currently pursued solution is to explore PennyLane / PyTorch ability to create hybrid models of well integrated quantum and classical components.

Such models' parameters are arranged into a series of layers that can be taken advantage of during the optimisation process.



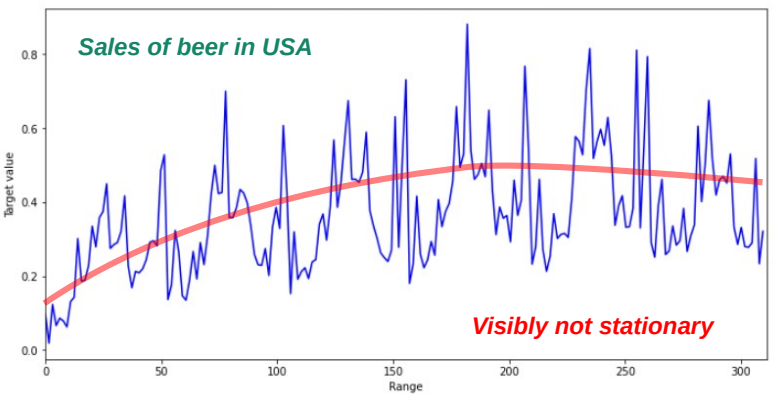
PennyLane / PyTorch Neural Network
of classical and quantum components

Thank you!

Any questions?

Appendix:
Concepts in TS Analysis

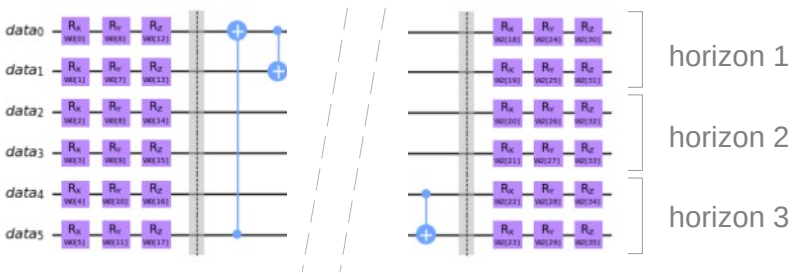
- Time series analysis aims to *identify patterns* in historical time data and to *create forecasts* of what data is likely to be collected in the future
- Applications include heart monitoring, weather forecasts, machine condition monitoring, etc.
- Times series analysis is well established with *excellent tools* and *efficient methods*, yet some organisations aim to improve them further
- Time series must have an *unique index* - a time-stamp sequencing the series
- Time series needs to be *ordered* by its index
- Time series will also have some *time-dependent attributes* to be modelled
- Time series can be *univariate* or *multivariate*, depending on whether a single or multiple attributes are being investigated
- *Missing indeces* and their dependent attributes may need to be imputed (e.g. interpolated)
- Index needs to be of appropriate *granularity*, e.g. years, months, weeks, days, hours, etc.
- Attributes need to be *aggregated* to the required index granularity
- Time signal often shows *seasonality* in data, i.e. a regular repeating pattern
- With aggregation and smoothing seasonality can be removed and *trends* visually identified
- Majority of forecasting methods require *time-series to be stationary*, i.e. its mean, variance and auto-correlation are constant
- *Quantum time series analysis (QTSA) is a promising approach to time series analysis and forecasting!*

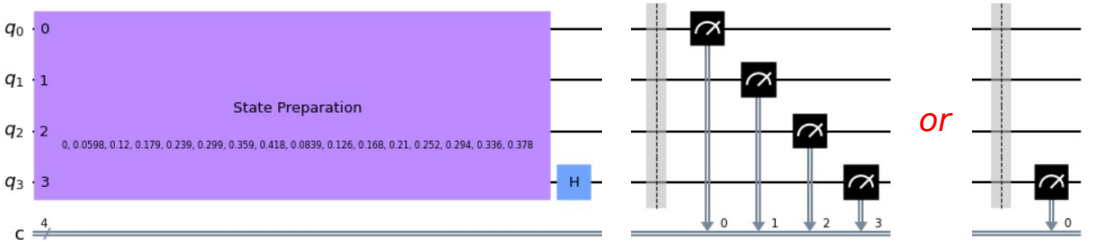


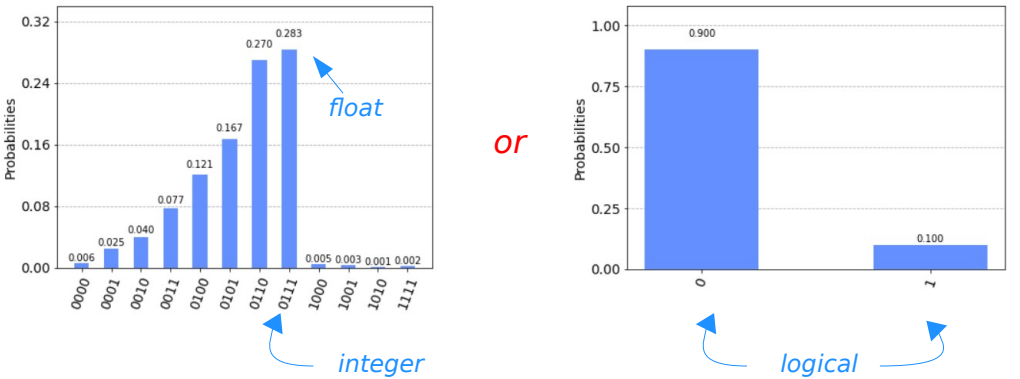
Appendix:
Measuring Quantum Circuits

There are many ways of obtaining the outcome of a circuit execution.

- We can select all qubits to measure
- We can select only those qubits that give you (theoretically) the most appropriate result
- We can interpret the counts of multiple measurements
- We can reinterpret circuit measurements into different combinations of outcomes, e.g. to predict larger QTSA horizons







Repeated circuit measurement can be interpreted as outcomes of different types, e.g.

- as a binary outcome
  (e.g. a single qubit measurement),
- as a bitwise representation of an integer number
  (e.g. most frequent combination of multi-qubit measurements), or
- as a value of a continuous variable
  (e.g. expectation value of a specific outcome).

Appendix:
Quantum Neural Networks

Abbas, Amira, David Sutter, Christa Zoufal, Aurelien Lucchi, Alessio Figalli, and Stefan Woerner. “The Power of Quantum Neural Networks.” Nature Computational Science 1, no. 6 (June 2021): 403–9. https://doi.org/10.1038/s43588-021-00084-1.

Schreiber, Amelie. “Quantum Neural Networks for FinTech.” Medium, May 8, 2020. https://towardsdatascience.com/quantum-neural-networks-for-fintech-dddc6ac68dbf.

- A typical QNN consists of two main components, i.e. a feature map and an ansatz (also called variational model)
- The feature encodes the input data and prepares the quantum system state, using as many features as there are qubits
- The ansatz consists of several layers and, similarly to a classical NN, is responsible for inter-linking the layers - this is accomplished by trainable Pauli rotation gates and entanglement blocks
- Finally, the qubit states are measured and interpreted as QNN output

Pattern Matching

- In contrast to function / data fitting, QNNs are able to perform pattern matching, i.e. work with a sequence of values themselves rather than with the mapping between an index and values
- In the following experiments, we will adopt a sliding window approach to structuring the time series
- However, the standard QNN model does not lean itself to time series analysis, i.e.
  - You are limited to the TS window of size equal to the number of qubits

In Qiskit VQR Model