The aims of this session:

To explore and understand encoding strategies for time series data

Introduction to QAEs
Denoising TS QAEs
Design choices
Architectural choices
Input encoding choices
Output / cost function choices
Encoder / decoder ansatze choices
Optimization / training choices
Qiskit vs PennyLane vs PyTorch
Summary of results
Conclusions and future work

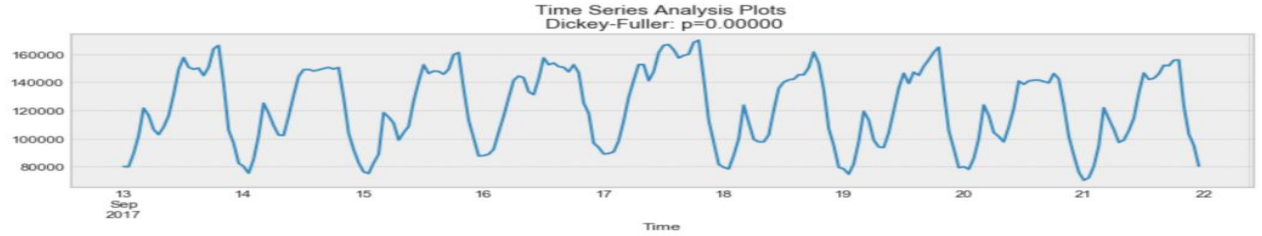
Data preparation for Quantum Autoencoders

Theoretical and practical, worst-case scenario: denoising time-series

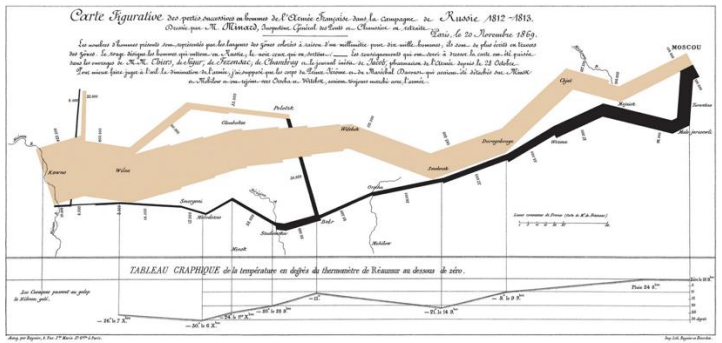
Jacob L. Cybulski
Enquanted, Melbourne, Australia

Sebastian Zając
SGH Warsaw School of Economics, Warsaw, Poland

15 March 2025, 5th Quantum Machine Learning Conference



Problem

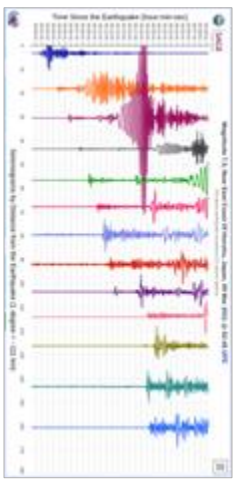


Napoleon's march in Russia 1812

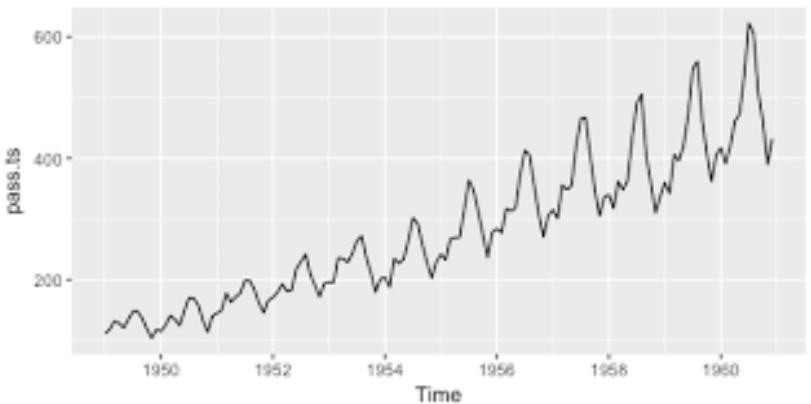
Time series analysis is a well-established math, stats, economics, and AI branch. Can quantum machine learning assist in detecting complex patterns in time series and signals from the preceding data sequences? How?

Sample applications: machine condition monitoring, astronomical observations, nationwide marketing and sales, earthquake prediction, and EEG or ECG analysis.

Earthquakes



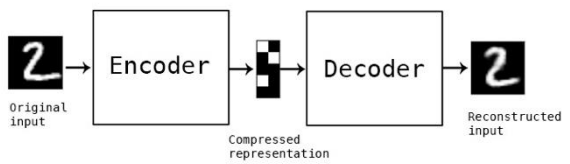
The Air passengers

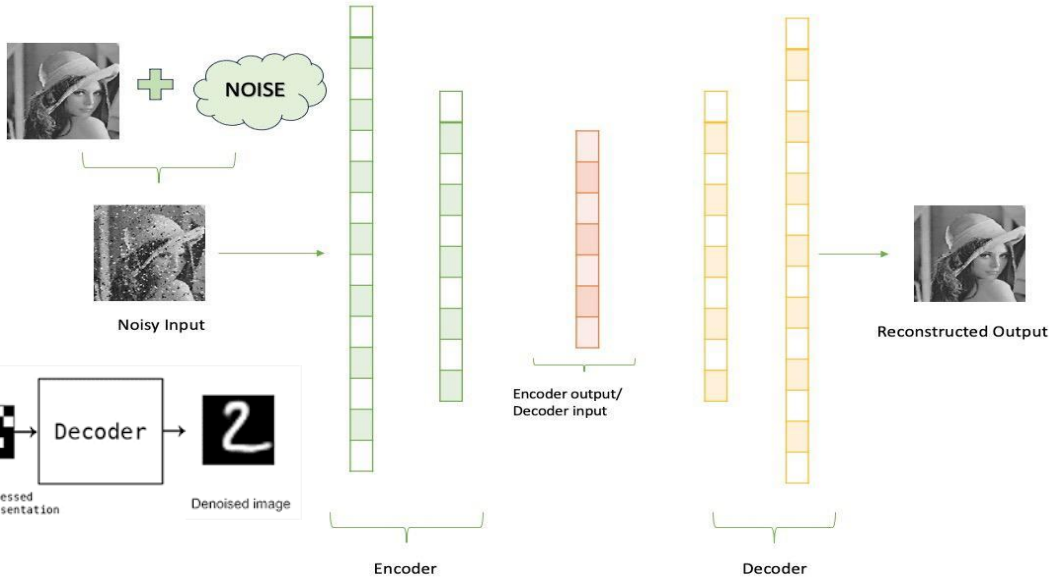


Autoencoder (as Deep Learning models)

Autoencoders are Deep Learning models that compress input into its essential features and then recover the original information from them.

AEs lose infrequent, insignificant, or unwanted information





Denoising AE is a slight modification to the vanilla AE that can reduce noise from real-world datasets.

Take noisy data points as input and reconstruct the clean version as output

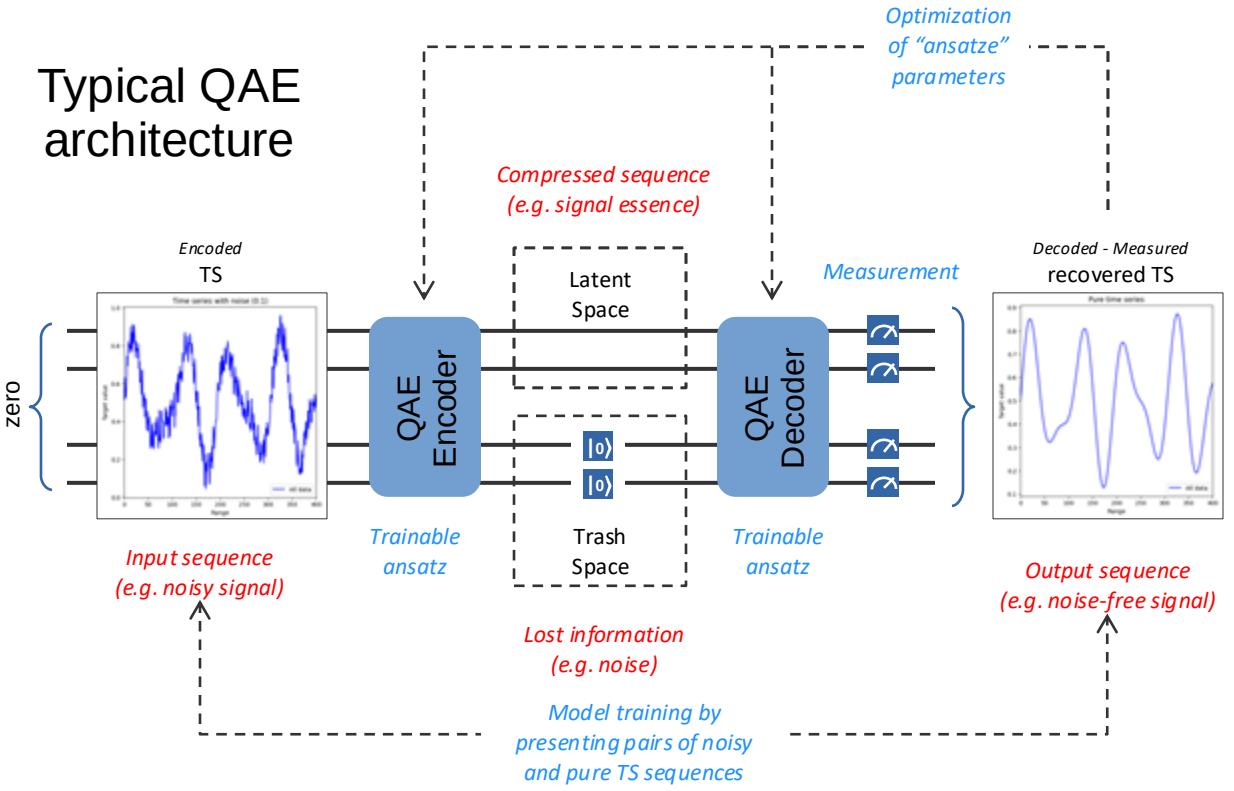
Quantum Autoencoder (for Time Series)

There are a few applications of QML methods to time-series analysis, TS applications of quantum AE (QAE) are even fewer

QAEs have the potential to deal with highly complex noise and anomaly patterns

Training of QAEs is difficult, due to:

- Potentially many features (e.g. TSs) (lots of qubits and/or parameters)
- Complex measurement strategies
- Unsupervised learning (we do not know what is noise)
- Possibility of barren plateaus



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

Simple problems can be solved with pure quantum methods
Complex issues require hybrid quantum-classical methods

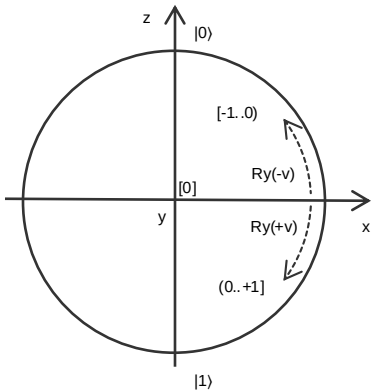
Input encoding / embedding
for QAE processing

In general, QAE input and output is an unrestricted collection of real values (floats) – this guided our selection of data encoding methods.

We rejected the following encoding methods:

Basis encoding, with qubits acting as bits in the encoded number (logical / int) to be processed later in the circuit.

QRAM encoding, where all possible inputs are known in advance, pre-coded in a circuit, and used by reference.



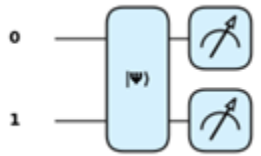
Angle encoding suits QAE design, with input values represented as qubit state rotations (float).

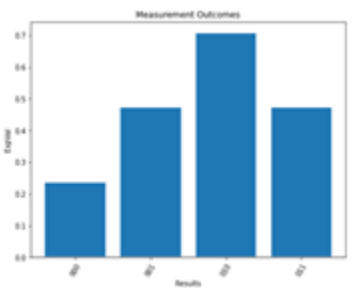
In our experiments we used angle encoding relative to $|+\rangle$ state, with values $\in [-1, 1]$ scaled to arange $[0, \pi]$, and coded as rotations up (<0) or down (≥0).

Amplitude encoding is probably the least understood, however, it is one of the most useful encoding schemes - attractive for QAEs.

It embeds input as a circuit state normally measured on output, i.e. each data point is encoded as expectation value *of multi-qubit measurement (int / float)*.

The problem with this encoding scheme is that for each unique input value, the structure of encoding gates is different. The circuit is not differentiable, which may be suitable for simulators, but difficult to use with GPUs and QPUs.



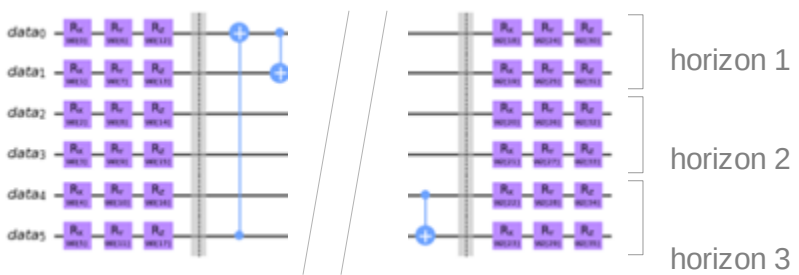


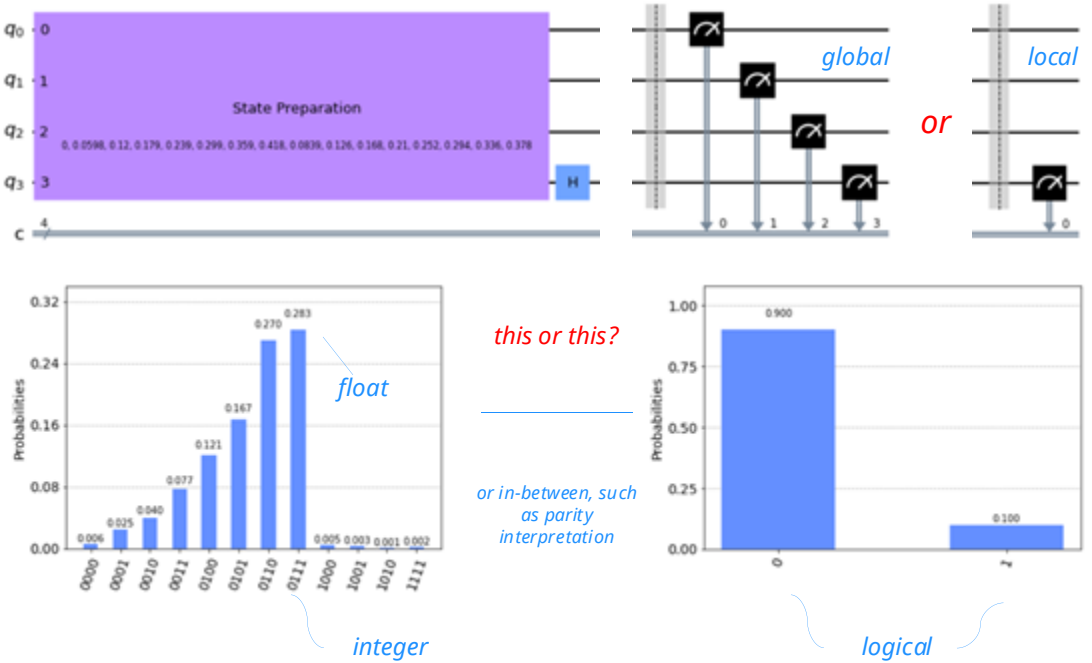
Example: data encoded as ψ was normalised vector [1/8, 2/8, 3/8, 2/8]. The measurement reflects the input data proportions.

Measurement & Interpretation

There are many ways of decoding the circuit state to form classical output data, e.g. we can:

- *measure all qubits*
  (as related to the global cost function)
- *measure a selection of qubits*
  (as related to the local cost function)
- *measure the circuit state in different ways*
  (e.g. as counts, expvals or probabilities)
- *reinterpret circuit measurements*
  into different combinations of outcomes, e.g. , to predict larger TS horizons (future)





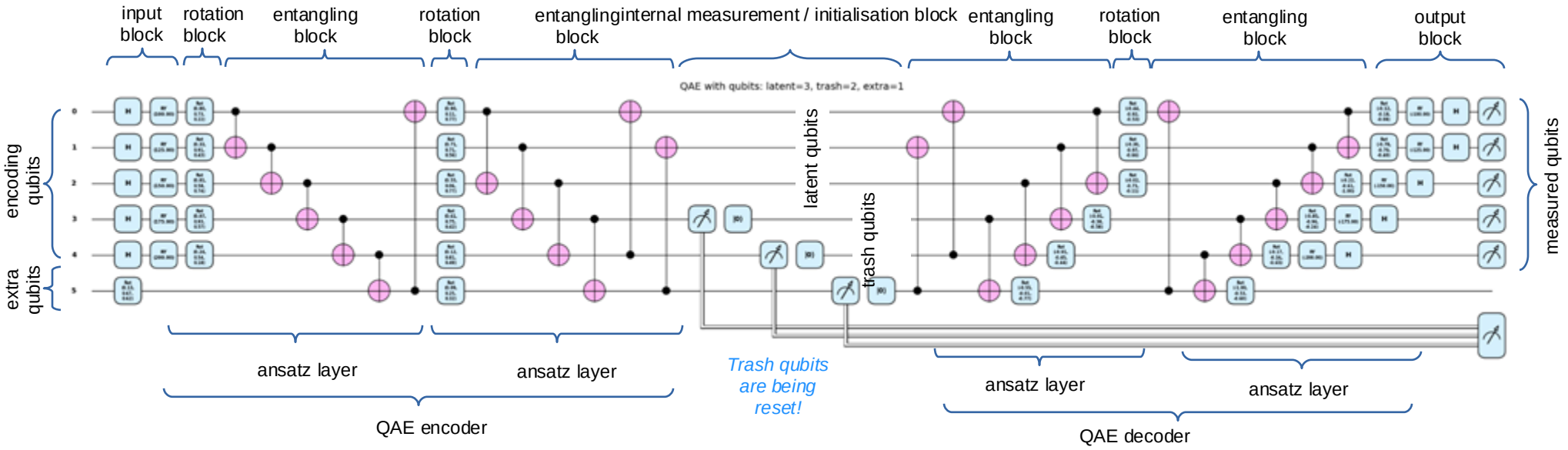
Repeated circuit measurement can be interpreted as outcomes of different numeric types, e.g. as a:

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

Anatomy of QAE Ansatze

QAE encoder and decoder (Qiskit)

This model features mid-circuit measurement which is not a unitary operation, hence it is not differentiable and hard to optimise.



QAE encoder and decoder are often symmetric (as shown here)

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

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

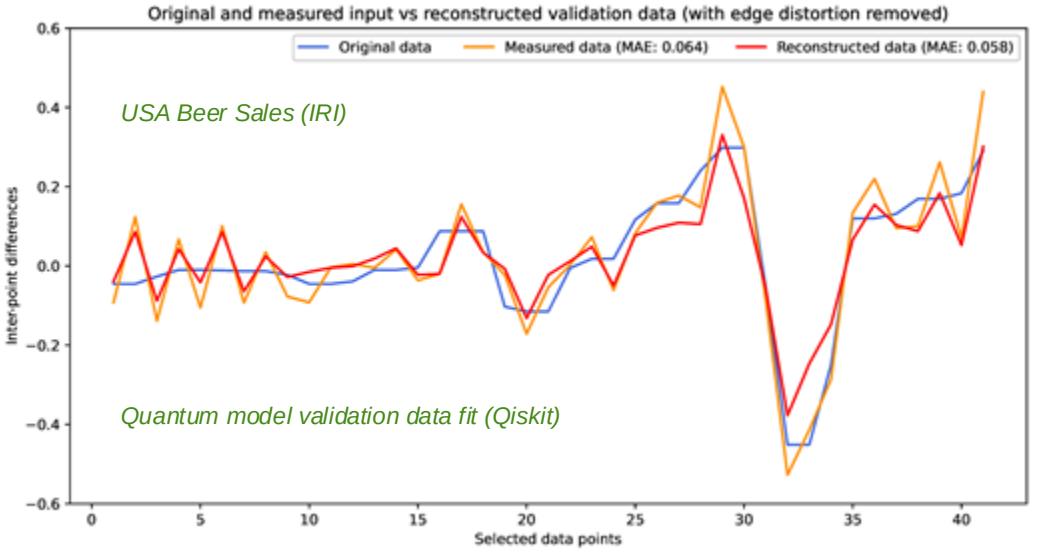
The selection of the optimizer of ansatze parameters requires some preliminary investigation of their effectiveness

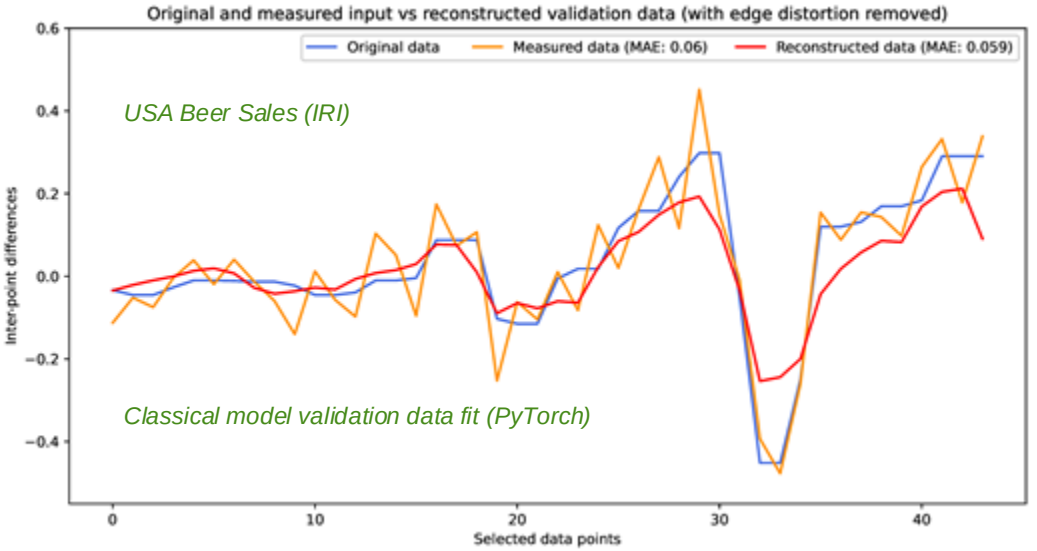
This depends on the model architecture, ansatz design, data encoding and decoding, as well as the nature of training data

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 QAE denoising TSs

The best QAE model's ability to remove noise from signals (left) was comparable to (but not better than) the best equivalent DL model (right).





In training, QAE models seem to learn (red) by reducing the amplitude between noisy signals (orange) and pure signals (blue), while retaining the shape of noise.

Classical CAE models seem to be better at fitting the intended signals shape (red), while reducing the amplitude between noisy signals (orange) and pure signals (blue).

On metrics, CAEs excelled in models training performance.
However, in validation CAEs and QAEs performed at the same level.

Problems discovered
Solutions proposed

Solution:
PennyLane/ PyTorch
approach to QAE development

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

One of the issues found to affect the QAE training performance was:

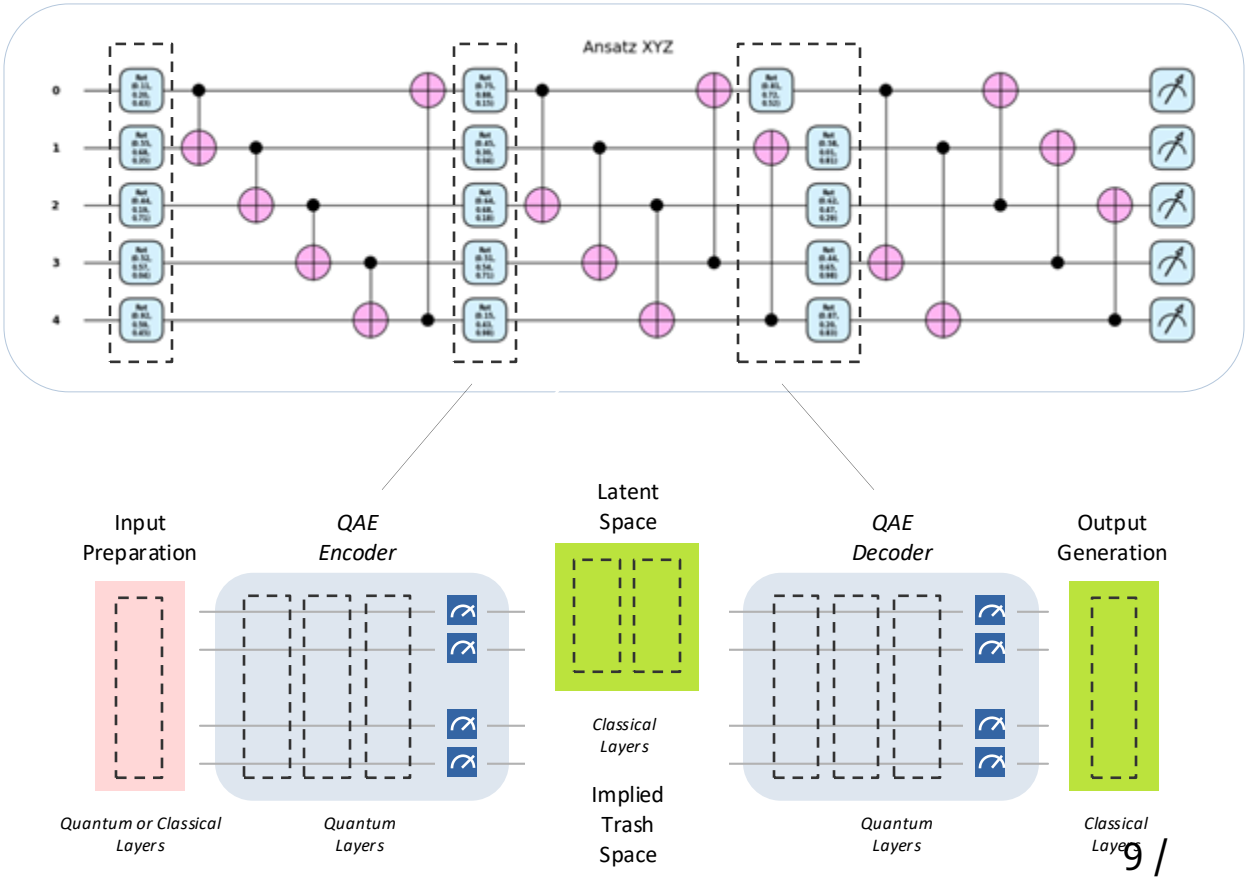
Dealing with deep quantum circuits consisting of large numbers of unstructured parameters

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

Large quantum models can be decomposed into classical DL NNs and several smaller quantum circuits.

Their parameters can be structured into layers so that PyTorch can manage them effectively during optimization.

Qiskit recently adopted a similar open source framework “torchquantum”.



Anatomy of QAE Ansatze

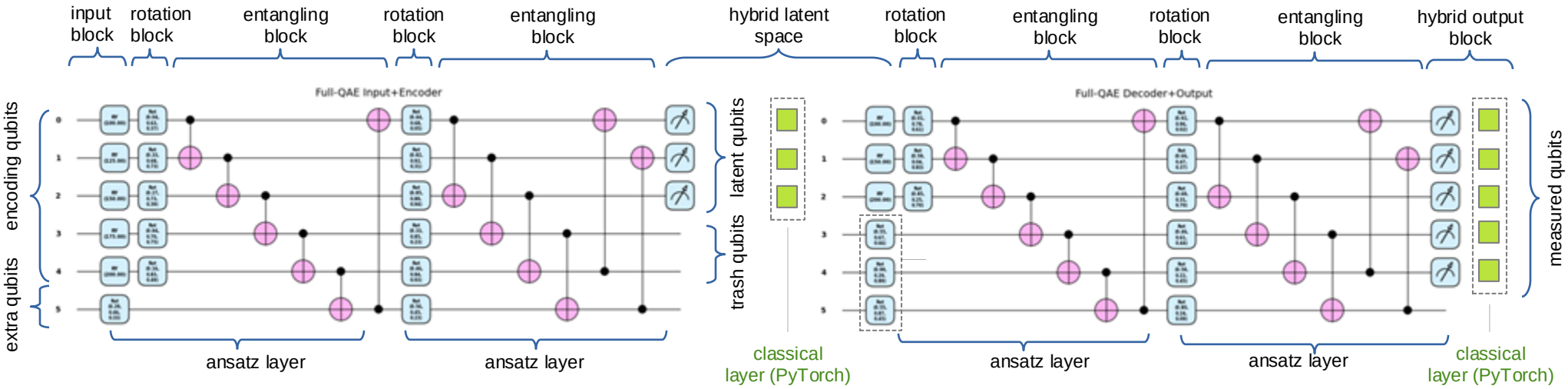
QAE encoder and decoder (PennyLane)

Note that classical layers are optional, however, they significantly improve the model performance when running on a quantum simulator.

They can also add features unavailable in pure quantum models (e.g. nonlinearity).

They may, however, prevent quantum advantage.

It is a “minimum” hybrid model



The QAE encoder and decoder do not need to be symmetric (and in this case, they are not). The hybrid QAE separates them into two shallow circuits, which can be trained effectively and quickly.

However, to the detriment of their function, hybrid QAEs lose some quantum information.

PennyLane and PyTorch offer excellent support for gradient manipulation, providing several highly efficient gradient optimizers. Therefore, we adopted a Nadam optimizer.

Note that Qiskit also supports passing gradients into its optimizers; however, this is not highlighted as a Qiskit feature.

Comparing results
(PennyLane vs PyTorch)

USA beer sales (IRI)

The experiments show:

The larger the QAE latent space, the better learning
(the accepted idea that reducing latent space helps abstraction is wrong)

There is an optimum depth for the QAE model.

PennyLane "minimum" hybrid models outperformed Qiskit models in training, but not in validation.

Within the limit of 1000 epochs, QAE outperformed CAE.

In general, QML models on simple tasks (such as DL AE) do not outperform the classical models – so to gain quantum advantage you need to pick the application very carefully.

Varying the latent space: DL CAE model in PyTorch @ 1000 epochs

	Experiments		Training					Validation				
	Lat	Tr	TR2	TMSE	TRMSE	TMAE	TMAPE	VR2	VMSE	VRMSE	VMAE	VMAPE
Run												
0	8	0	0.9621	0.0087	0.0925	0.0716	0.0615	0.7475	0.0478	0.2105	0.1583	0.1444
1	7	1	0.9636	0.0086	0.0925	0.0683	0.0607	0.7491	0.0463	0.2016	0.1614	0.1431
2	6	2	0.9631	0.0081	0.0911	0.0708	0.0604	0.7547	0.0466	0.2133	0.1554	0.1443
3	5	3	0.9592	0.0085	0.0925	0.0710	0.0624	0.7468	0.0455	0.2133	0.1633	0.1467
4	4	4	0.9609	0.0088	0.0941	0.0713	0.0618	0.7668	0.0445	0.2120	0.1604	0.1445
5	3	5	0.9625	0.0092	0.0973	0.0701	0.0646	0.7461	0.0453	0.2146	0.1677	0.1464
6	2	6	0.9515	0.0121	0.1096	0.0815	0.0697	0.7144	0.0516	0.2264	0.1694	0.1504
7	1	7	0.8575	0.0321	0.1788	0.1274	0.1098	0.4706	0.0937	0.3012	0.2217	0.1859

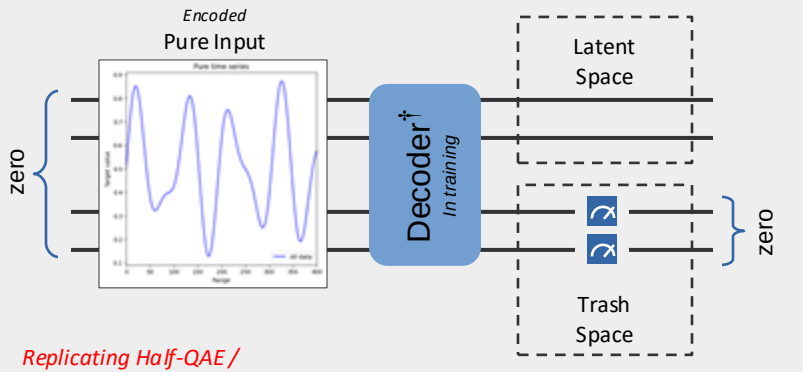
Varying the circuit depth: quantum model in PennyLane + PyTorch @ 1000 epochs

	Experiments				Training					Validation				
	Lay	Lat	Tr	Xtr	TR2	TMSE	TRMSE	TMAE	TMAPE	VR2	VMSE	VRMSE	VMAE	VMAPE
Run														
8	1	5	3	0	0.7663	0.0449	0.2112	0.1508	0.1285	0.1460	0.1019	0.3154	0.2192	0.2081
9	2	5	3	0	0.9635	0.0084	0.0910	0.0703	0.0652	0.6278	0.0475	0.2169	0.1656	0.1598
10	3	5	3	0	0.9589	0.0093	0.0953	0.0693	0.0631	0.6926	0.0400	0.1994	0.1470	0.1397
11	4	5	3	0	0.9644	0.0081	0.0885	0.0656	0.0592	0.6890	0.0413	0.2028	0.1545	0.1457
12	5	5	3	0	0.9572	0.0096	0.0971	0.0693	0.0624	0.7198	0.0386	0.1962	0.1474	0.1374
13	6	5	3	0	0.9528	0.0104	0.1015	0.0722	0.0642	0.6915	0.0408	0.2016	0.1531	0.1445
14	7	5	3	0	0.9499	0.0111	0.1052	0.0747	0.0659	0.6866	0.0412	0.2027	0.1502	0.1404
15	8	5	3	0	0.9525	0.0106	0.1027	0.0728	0.0649	0.7073	0.0400	0.1999	0.1503	0.1411

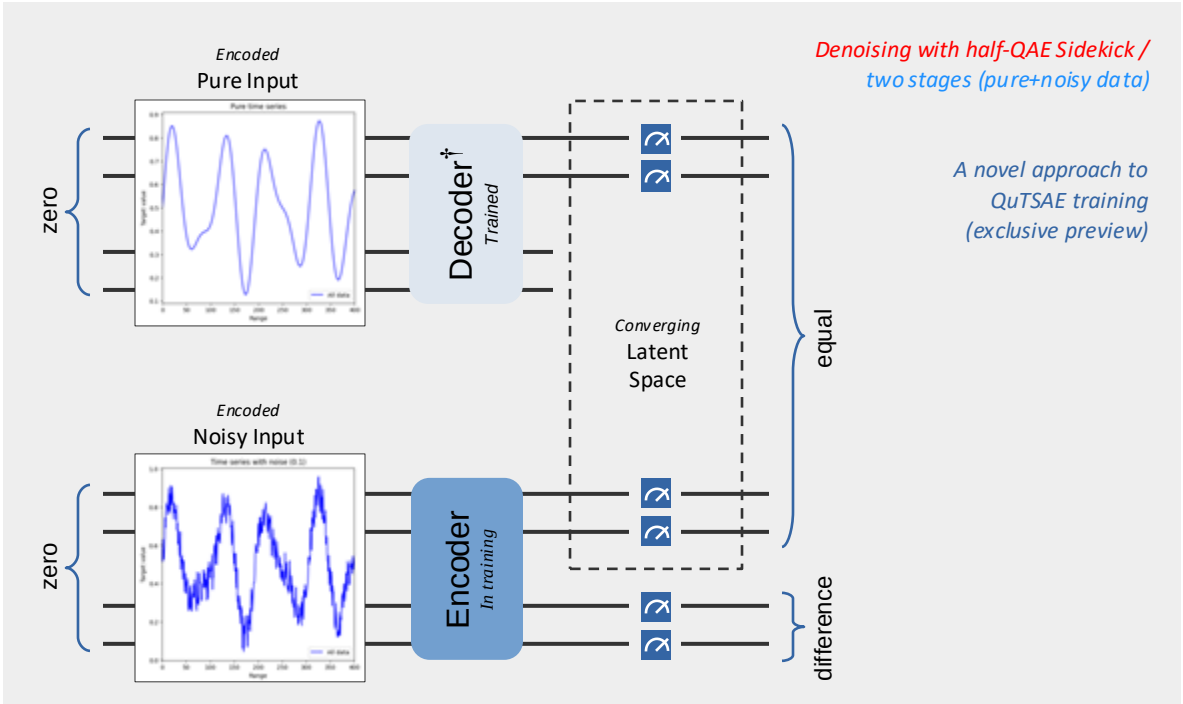
Varying the latent space: quantum model in PennyLane + PyTorch @ 1000 epochs

	Experiments				Training					Validation				
	Lay	Lat	Tr	Xtr	TR2	TMSE	TRMSE	TMAE	TMAPE	VR2	VMSE	VRMSE	VMAE	VMAPE
Run														
0	3	8	0	1	0.9732	0.0062	0.0770	0.0581	0.0541	0.7139	0.0417	0.2034	0.1545	0.1478
1	3	7	1	1	0.9736	0.0061	0.0764	0.0579	0.0545	0.7350	0.0373	0.1928	0.1467	0.1460
2	3	6	2	1	0.9667	0.0076	0.0864	0.0643	0.0602	0.6953	0.0438	0.2083	0.1518	0.1463
3	3	5	3	1	0.9540	0.0103	0.1003	0.0731	0.0653	0.6770	0.0455	0.2126	0.1620	0.1499
4	3	4	4	1	0.9244	0.0160	0.1221	0.0879	0.0765	0.6189	0.0499	0.2211	0.1688	0.1593
5	3	3	5	1	0.9056	0.0194	0.1346	0.0980	0.0866	0.6106	0.0553	0.2332	0.1765	0.1642
6	3	2	6	1	0.8435	0.0309	0.1703	0.1205	0.1035	0.4838	0.0653	0.2533	0.1814	0.1662
7	3	1	7	1	0.7197	0.0522	0.2284	0.1521	0.1263	0.2278	0.0895	0.2991	0.2136	0.1878

Alternative Architectures



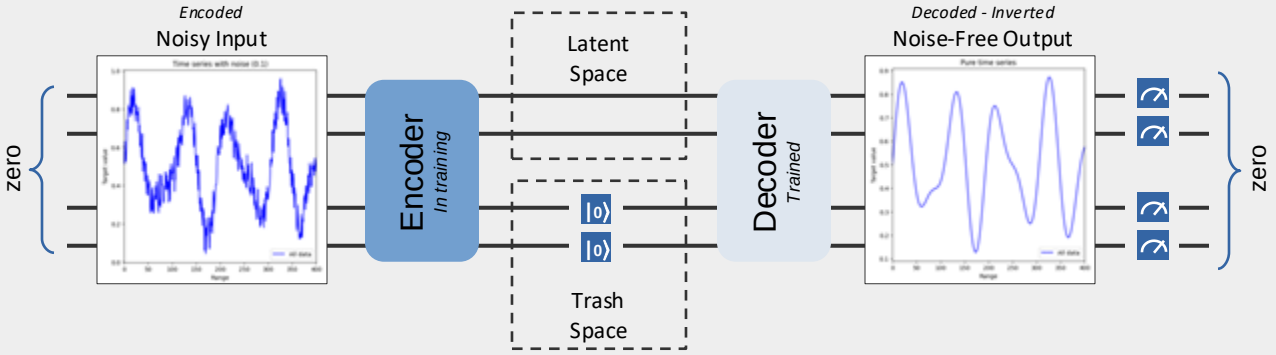
Replicating Half-QAE /
single stage (pure data only)



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

A novel approach to QuTSAE training (exclusive preview)

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



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

Summary

Simulated VQA QAE models

Model design insights

- We have discussed design decisions taken in the development of denoising quantum time series autoencoders.
- Input encoding strategy determines what ansatz can be employed, and vice versa.
- Methods of measuring and interpreting a quantum state impact the choice of the loss/cost function.
- Ansatz architectural properties must fit the model's aim and function
- Ansatz width, depth, the number of trainable parameters, additional degree of freedom (extra qubits), and data used in training, all influence the success of the model optimization
- Selection of a suitable cost function and an optimiser requires experimentation

Model quality and performance insights

- Assessment of the quantum model quality requires a suitable theory and statistical analysis!
- QAE pure quantum models are merely approaching the performance of classical models
- On simulators, hybrid quantum-classical models perform better than either pure quantum or pure classical models
- Hybrid models lose their quantum efficiency when trained and executed on quantum machines
- Quantum models can take advantage of the model features absent from classical systems
- Hybrid models can inject the model features missing from the pure quantum systems

Current and future work

- Tighter integration between classical and quantum methods – more effective optimisation
- Investigation of different QAE architectures,
- Moving beyond noise – anomalies and chaos
- Moving beyond QAEs – QGANs and QTransformers

Thank you!