The aims of this session:

To explore and understand Quantum Autoencoders and their various designs

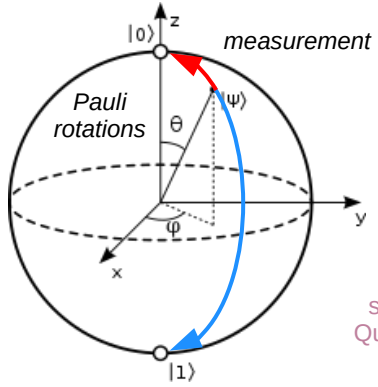
To undertake a brave, hands-on, and independent exploration of pure and hybrid quantum-classical QAEs

Introduction to QAEs
Denoising TS QAEs
QAE architectural choices
QAE input encoding choices
QAE output / cost function choices
QAE encoder / decoder ansatze
QAE optimization / training
QAE training with noise
Conclusions
Challenge tasks

Quantum Autoencoders

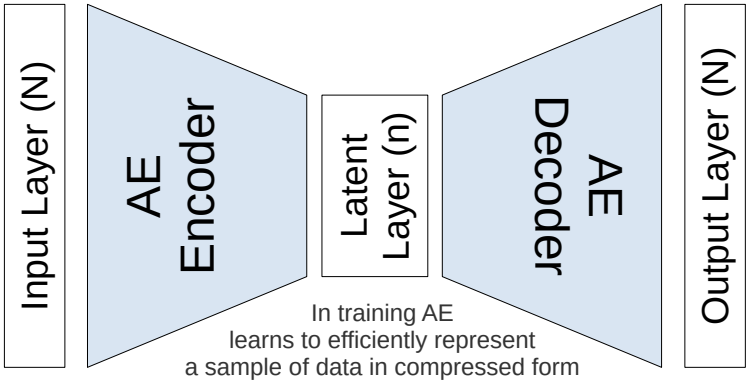
Applications of QAEs to time series analysis
Advanced session with some instruction

Jacob L. Cybulski
Enquanted, Melbourne, Australia



We will assume
some knowledge of
Quantum Computing
ML and Python

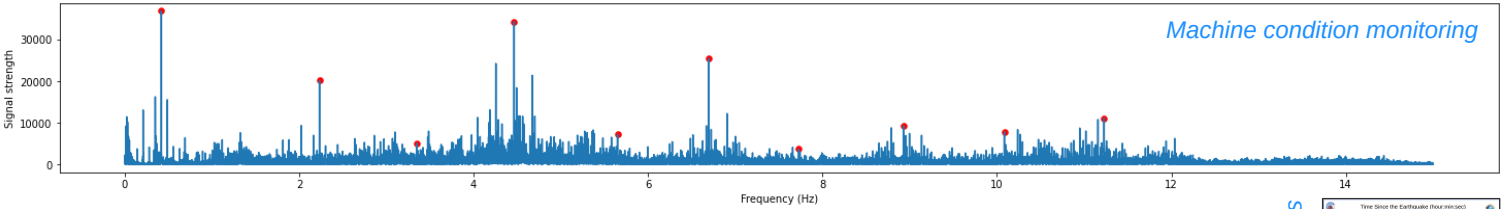
Classical Autoencoders & Quantum Autoencoders



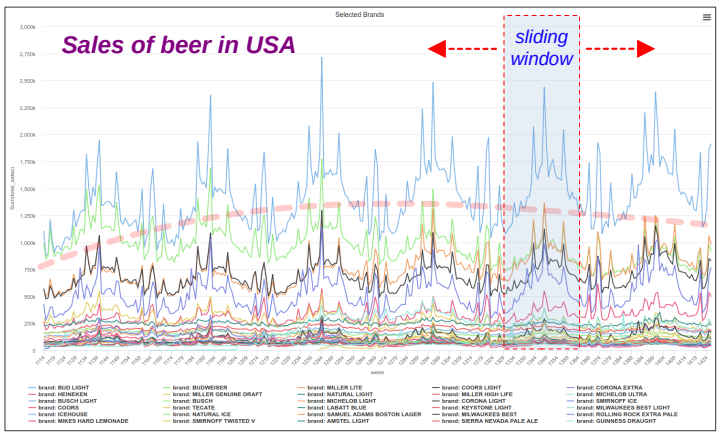
- ***Autoencoders (AE)*** are deep learning (DL) models that encode information into a compact / compressed form, from which approximate information content can be decoded effectively
- In the process AEs lose the infrequent, insignificant or unwanted parts of information
- Typically, an AE is implemented as a multilayer perceptron, which includes:
  - *Input layer* of *N* nodes of some data
  - *AE encoder* consisting of several neural network layers mapping (encoding) input of *N* nodes into a smaller layer of *n* nodes (n < N)
  - *Latent layer* (also known as “code”) containing efficient representation (compressed) of input
  - *AE decoder* consisting of several neural network layers recreating (decoding) input information
  - *Output layer* of N nodes representing decoded (decompressed) information
- Used for data compression, representation, data search, denoising and anomaly detection, e.g. in images and signals

- ***Quantum Autoencoders (QAE)*** utilise quantum machine learning methods to implement AE models
- There are still few practical applications of QAEs
- QAEs have the potential to remove highly complex noise and anomaly patterns
- QAEs can generalise data from latent space
- Training of QAEs is considered difficult, due to:
  - *Semi-supervised learning* (emergent latent representation)
  - *Highly dimensional models* (qubits, layers, params)
  - *Complex measurement strategies* (e.g. SWAP tests)
  - *Barren plateaus* potentially emerging in training
- We have great interest in QAEs because some of their design aspects cannot be replicated with any classical methods!

QAEs for
TS denoising



- *Time series (TS) and signal analysis* aim to identify patterns in historical time data and to create forecasts of what data is likely to be collected in the future
- TS *applications* include heart monitoring, weather forecasts, machine condition monitoring, etc.
- Time series can be *univariate* or *multivariate*
- Time series often show *seasonality* in data, i.e. patterns oscillations = repeating over time



Mean is not constant

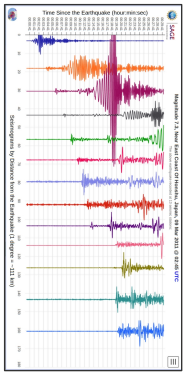
Variance is not constant

Trend is non-linear

It is not stationary

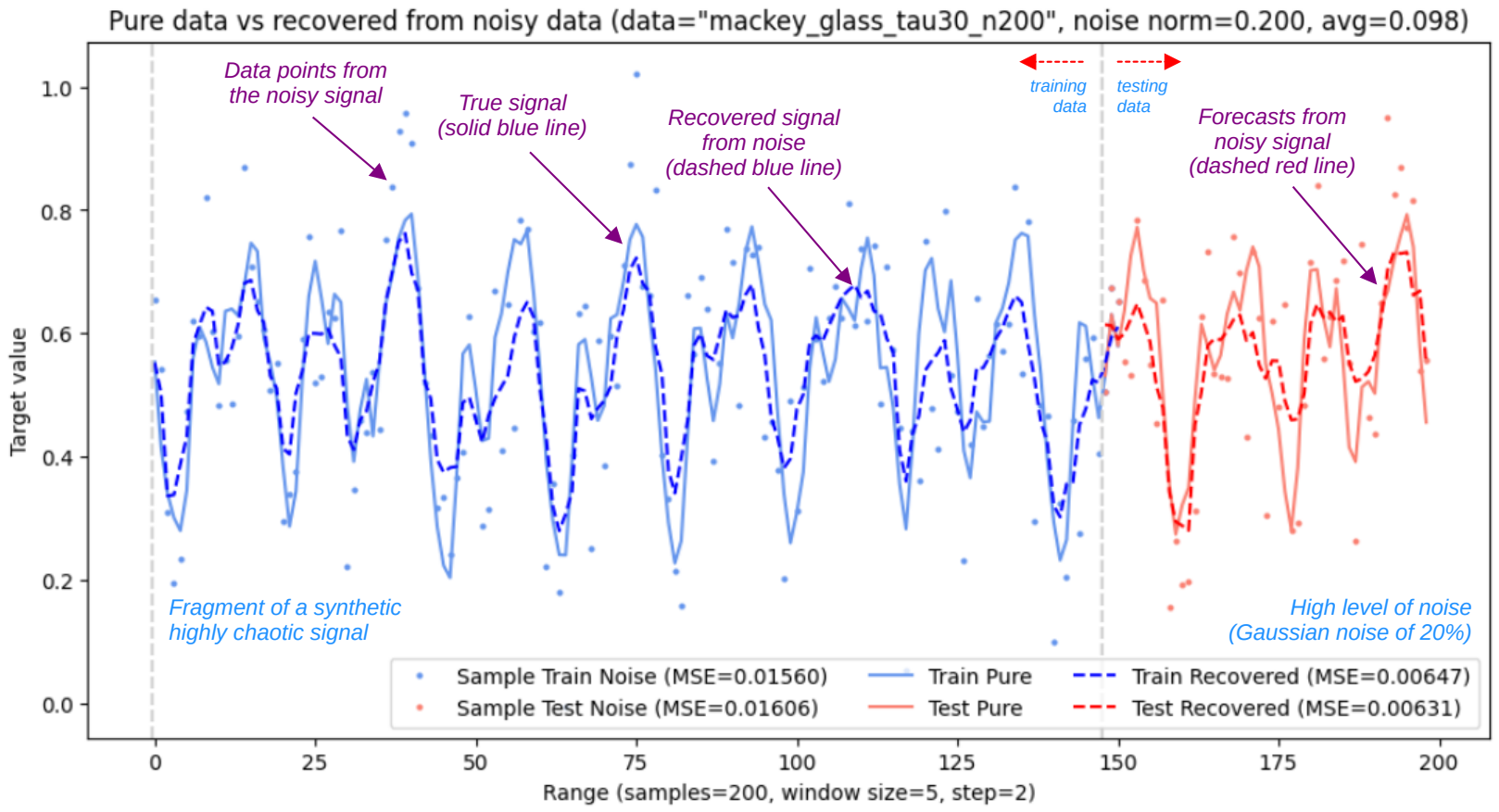
- *Challenges* of quantum TS analysis:
  - There is a *high volume of data*
  - TS *data ages* quickly
  - TS *values are not independent*
  - Consecutive TS *values are homogenous*

Earthquakes



- *Benefits* of quantum time series models:
  - Statistical TS analysis requires *strict data preparation*, such as dimensionality reduction, aggregation, imputation of missing values, removal of noise and outliers, adherence to normality and homoskedasticity, and they need to be stationary
  - *QML methods are more flexible* and do not have such strict requirements, hence QAEs are promising for effective analysis of complex temporal data!
- *QAE reduction of noise in signals* is the primary objective of this session
- *Sliding window protocol* will be used to analyse a small TS sub-sequence at a time

Problem and a possible solution



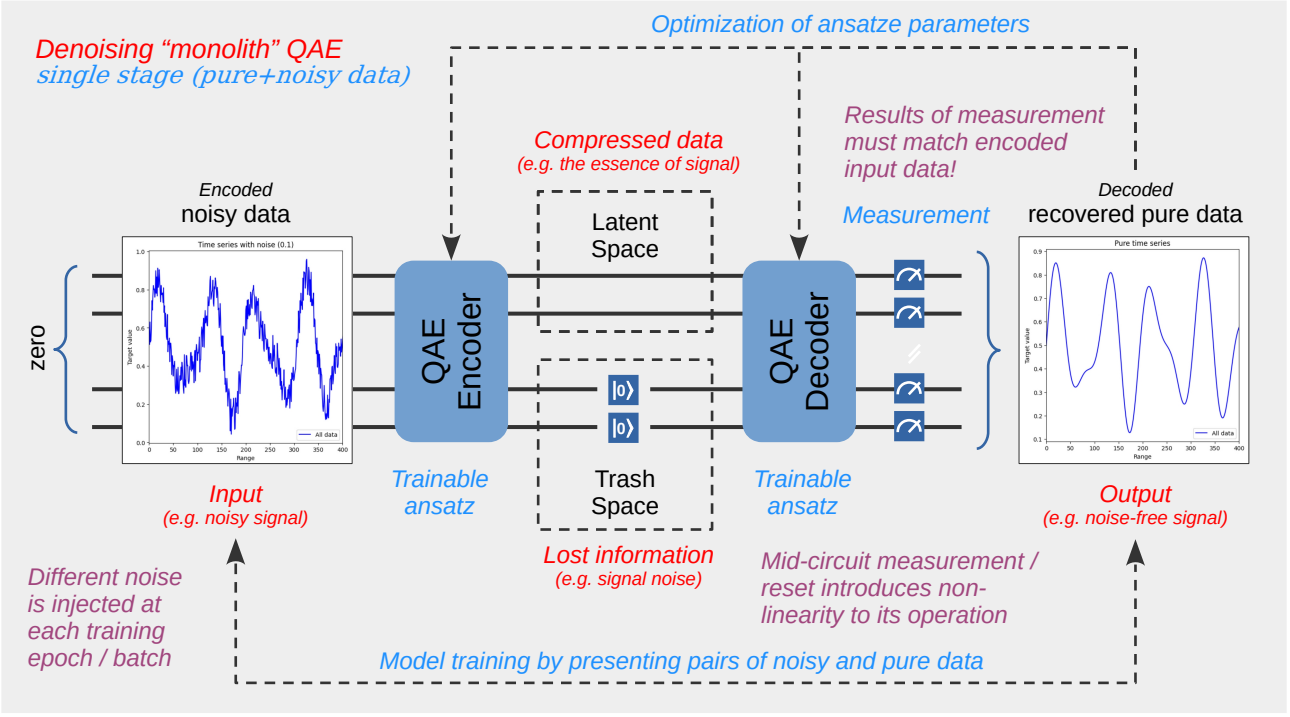
Question:
should we average inaccurate forecasts or should we try a model based denoising of forecast inaccuracies?

Another question:
what is normal and what is abnormal?

We will develop a TS QAE to remove high levels of noise from signals (past and future)

TS QAE implementation in PennyLane

The “monolith” architecture
of TS denoising QAE



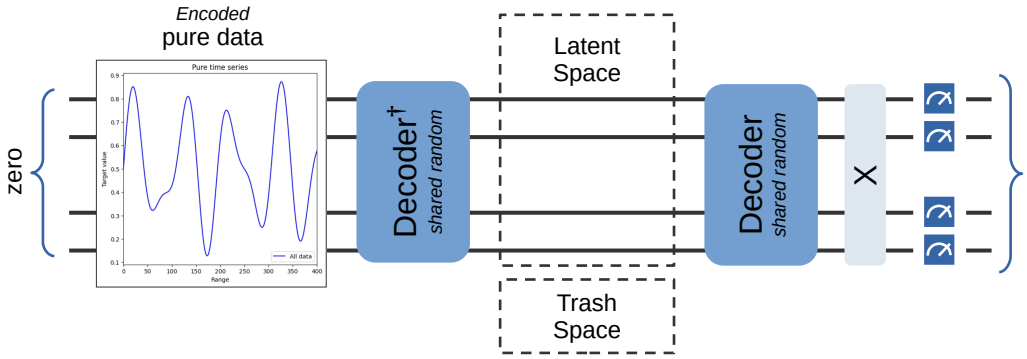
As the “monolith” QAE could potentially have lots of weights, its training can be computationally very expensive. We can cleverly reorganise the process of circuit optimisation by training the QAE encoder and QAE decoder separately. By doing so, we can potentially gain in training speed, however, likely reducing the recovery accuracy.

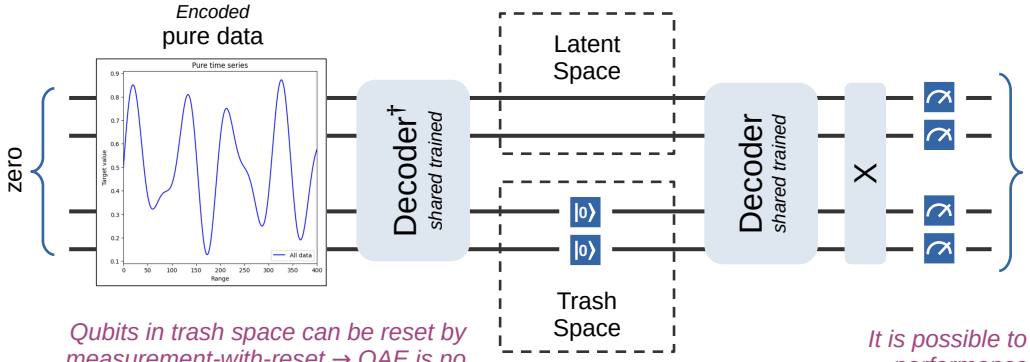
QAEs are quantum models made of the following components:

- *Quantum model:* a circuit of *N* qubits to match time series windows of *N* values - this may vary depending on the QAE architecture
- *Input encoder:* a quantum feature map embedding a window of noisy classical TS values on input, thus preparing the circuit state
- *QAE encoder:* a quantum ansatz of several layers consisting of trainable parameter blocks and entangling blocks, evolving the state of *N* qubits into a state of *n* qubits (n < N)
- *Latent space:* representing the essential features of the window embedded in the initial circuit state
- *Trash space:* representing information lost in QAE training, such as signal noise, it is measured and reinitialised to prevent flow of this information to the QAE decoder
- *QAE decoder:* a quantum ansatz of several layers consisting of trainable parameter blocks and entangling blocks, evolving the state of the latent space of *n* qubits into a state of *N* qubits
- *Output layer:* a measurement block resulting in classical data, which can be interpreted as a TS window of *N* values with reduced noise

Replicating QAE with half-QAE

Consider an angle encoding TS QAE consisting of encoder and decoder unitaries that have mirror image structures, i.e. a QAE Decoder and its adjoint QAE Decoder† used as QAE Encoder.





Qubits in trash space can be reset by measurement-with-reset → QAE is no longer unitary; alternatively with the SWAP operation → QAE stays unitary

It is possible to improve the replication performance by breaking the weight symmetry, however, we are no longer able to rely on the half-QAE training

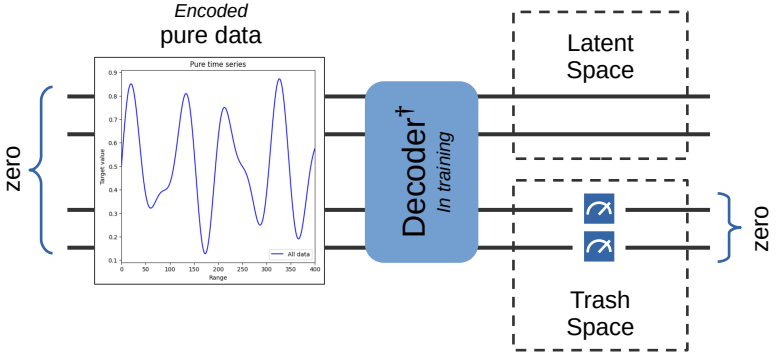
When the latent space spans all qubits, there is no loss of information, and so the QAE is a unitary.

As the QAE Encoder is an inverse of the QAE Decoder, they cancel each other operations.

This means that on output the QAE will always produce an adjoint of its input (upside-down TS).

In angle encoding, this can be corrected with a block of X operations on each qubit.

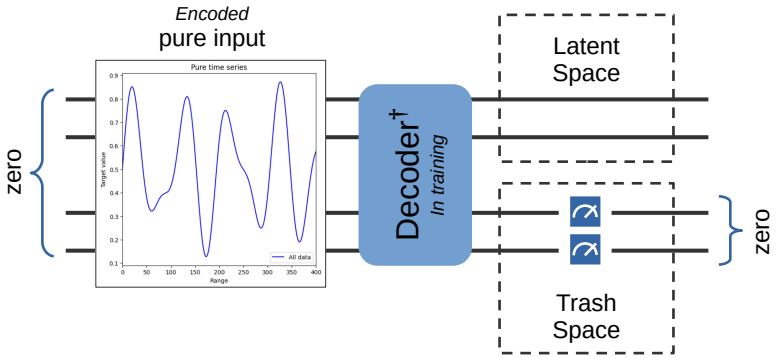
When we reset trash space, we start losing information flowing from the encoder to decoder.



However, we can train the QAE Encoder / Decoder to reduce this information loss, while preserving their mirror structure and symmetric weights.

As QAE Encoder is an adjoint of QAE Decoder, training only QAE Decoder† is enough, e.g. by ensuring most of its information flows through the latent space, i.e. by converging trash to zero.

Denoising TS with Stacked half-QAEs

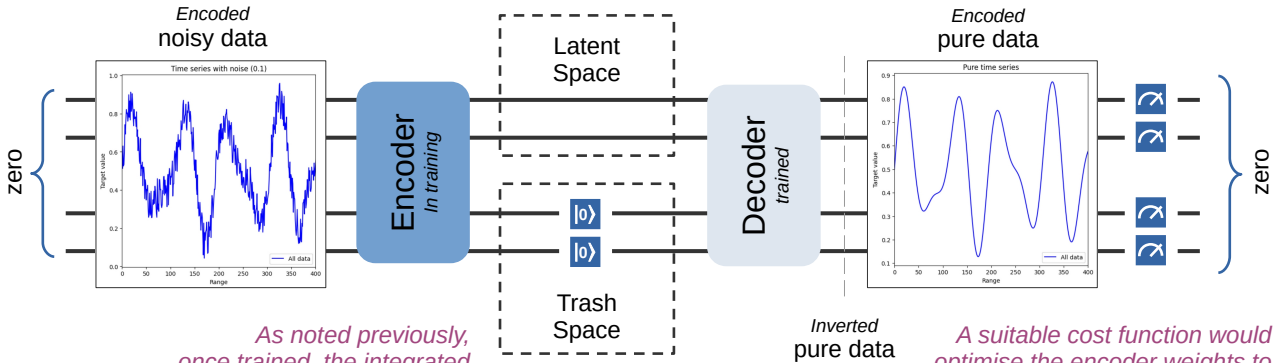


There is a widely held belief that replicating QAEs, which can be trained by using only their QAE Decoder, are capable of reducing the signal on input to its most essential information and remove all infrequent, noisy or inessential information.

This means that by feeding a noisy signal on input it would be possible to filter the noise out, retaining its pure component.

However, practice shows that replicating QAEs perform poorly in noise reduction.

An alternative is to use all QAE weights to their full extent.



As noted previously, once trained, the integrated circuit will require to invert its output with X on every qubit

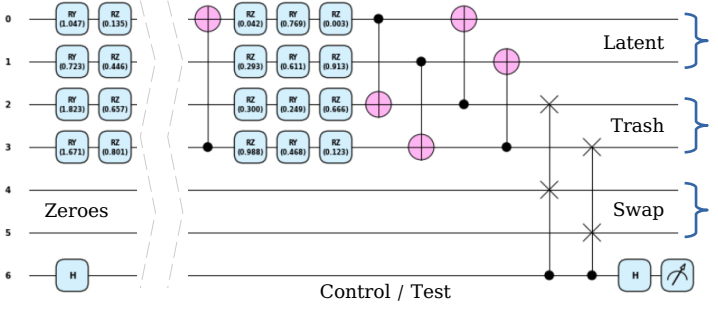
A suitable cost function would optimise the encoder weights to converge output to zero

We will train the denoising QAE in two phases.

Phase 1: Training inverted QAE Decoder using pure data and a cost function aiming to converge trash to zero. A common way of constructing such a cost function is to use a SWAP test.

SWAP test ensures QAE stays unitary, this means differentiable using the adjoint method!

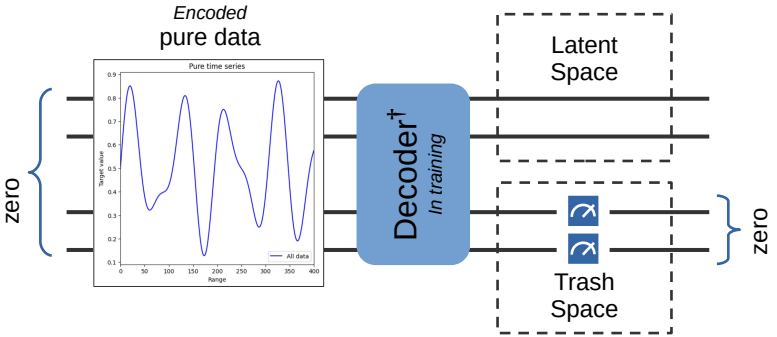
SWAP test conditionally switches the state of trash qubits with zero initialised qubits. Phase kickback allows monitoring the state of the qubits switching their states. Probability measurement of the control qubit of 1.0 indicates qubits to be identical, while 0.5 meaning them to be completely different (can translate to expval results).

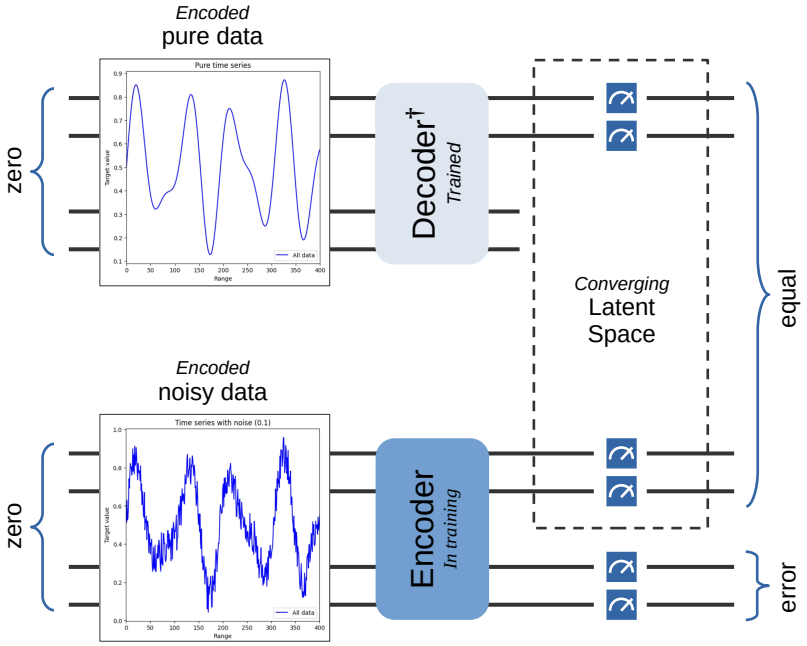


Phase 2: Training a QAE Encoder with noisy data, in combination with an adjoint of the previously trained QAE Decoder, will produce an inverse of pure data approximation. In a perfectly functioning QAE this output would cancel pure data encoding, which can be measured as zero.

Denoising TS with Sidekick half-QAEs

We can train the QAE encoder and decoder using variety of cost functions. For example, both the "monolith" and "stacked" configurations, rely on the similarity of the predicted and expected values. However, here we present something very different.





Warning: as the "sidekick" cost function never refers to the actual data, data encoding must correctly match the reintegrated model's measurements!

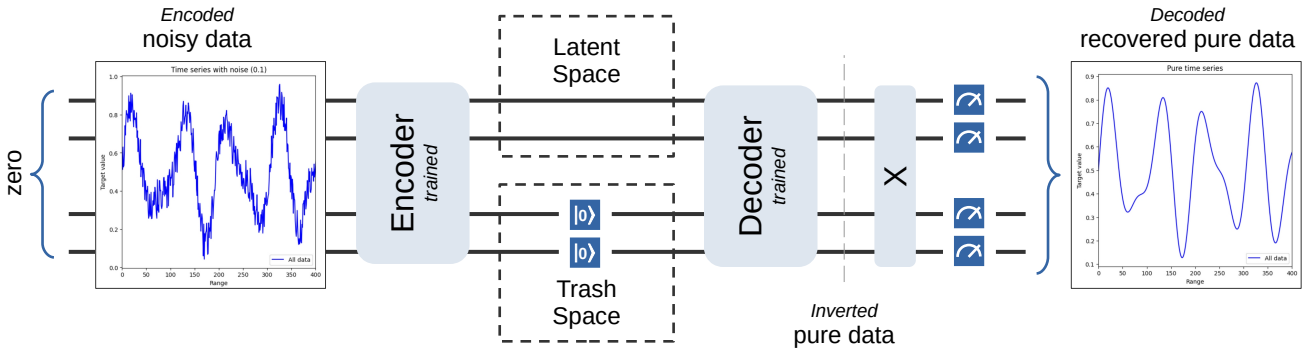
Once the inverted QAE decoder is trained on zero trash, the decoder can assist the QAE to be trained in a parallel structure. The decoder becomes a "sidekick".

In this scenario, the convergence of two half-QAEs can be achieved on the convergence of their latent spaces.

Again, a SWAP test can be used for this purpose.

In all those cases, where we initiate model training with an adjoint of a decoder converging on zero trash, when we integrate all QAE components, we need to invert the final output before measurement.

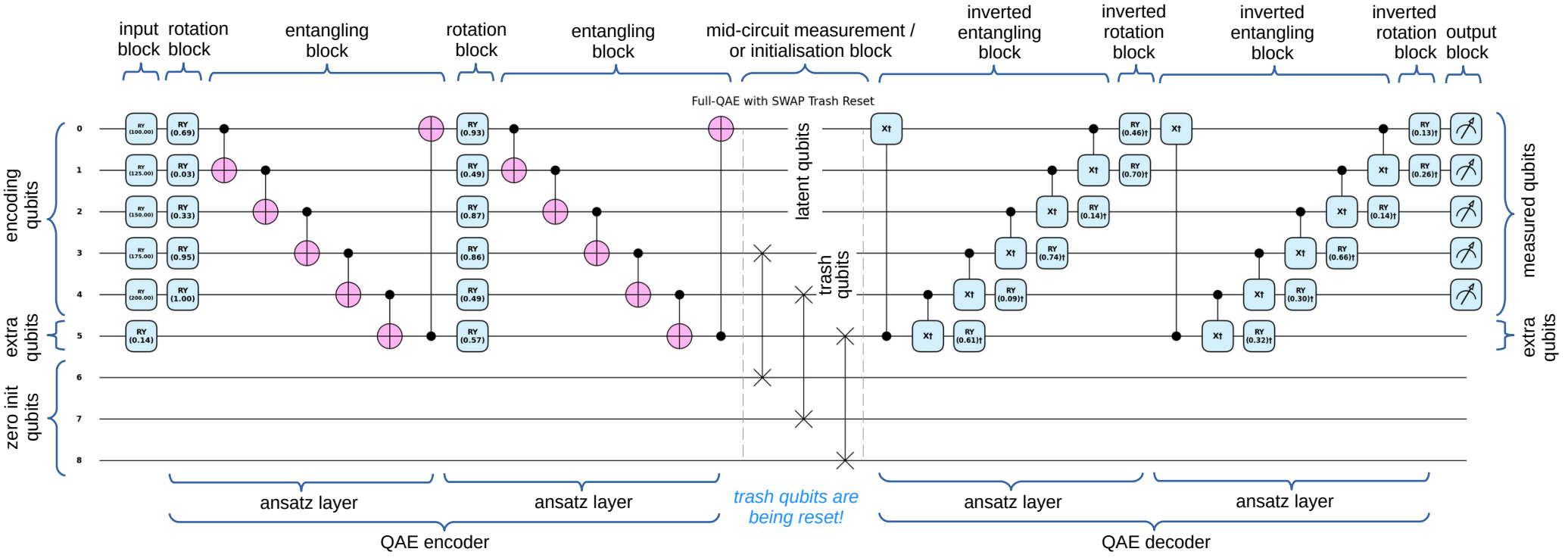
In fact, this is also applicable to all QAEs with a symmetric architecture, as such a structure is would likely improve the model training.



Anatomy of QAEs

QAE encoder and decoder

If mid-circuit measurement is used to reset trash qubits, the circuit is no longer unitary, not differentiable, and slow to optimise. Instead we can apply SWAP operations with zero initialised qubits, as shown here. When the circuit uses few qubits, PennyLane replaces mid-circuit measurement with SWAP.



QAE encoder and decoder are often symmetric.

They are layers of parameterised rotation blocks (e.g. Rx, Ry, Rz) and entangling blocks (e.g. Cx).

Ansatze may be extended with extra qubits to create more trainable parameters.

The selection of the optimiser 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.

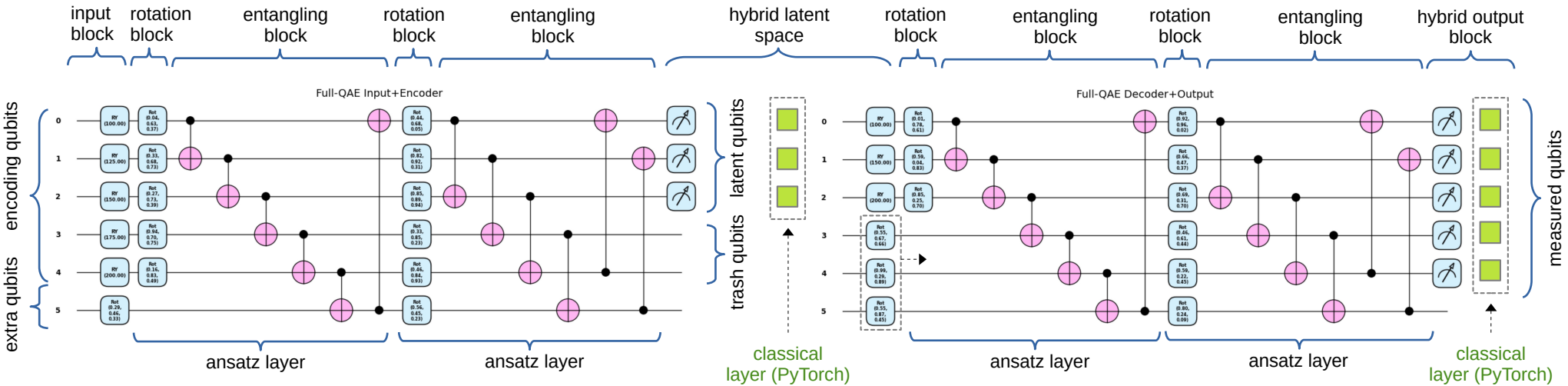
In our project we evaluated gradient based optimisers (e.g. ADAM).

Anatomy of Hybrid QAEs

Hybrid QAE encoder and decoder

Note that classical layers are optional, however they greatly improve the model performance when running on a quantum simulator. They can also add features not available in pure quantum models (e.g. nonlinearity). However, they may prevent effective utilisation of quantum hardware.

Here is a “minimum” hybrid model...



Training of the “monolith” QAE always faces difficulties due to the large dimensionality of its parameter space. This was partially addressed by training its half-QAEs separately.

Al alternative strategy is to adopt a hybrid QAE architecture, which is organised its into a combination of classical layers and *shallow quantum layers*, trained efficiently together.

However, in the process of mid-circuit measurement, *hybrid QAEs lose phase information* to the detriment of their function and effectiveness = possible quantum advantage.

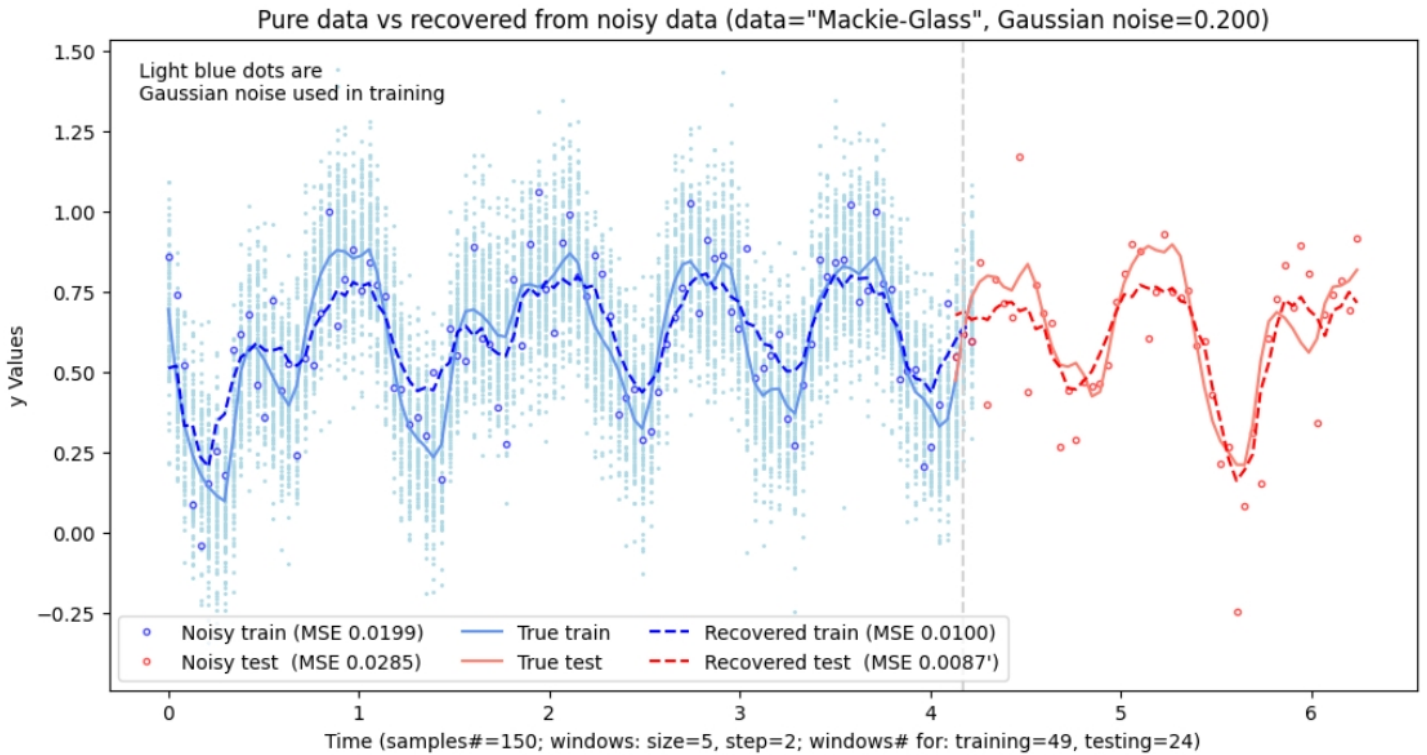
QAE encoder and decoder do not need to be symmetric, e.g. here, they are not mirror images of each other.

PennyLane and PyTorch have excellent support for *manipulation of gradients*, offering several highly efficient gradient optimisers. For example, here we can adopt an *NAdam optimiser*.

Note that other quantum SDKs, such as Qiskit, also provide great gradients support used by their optimisers.

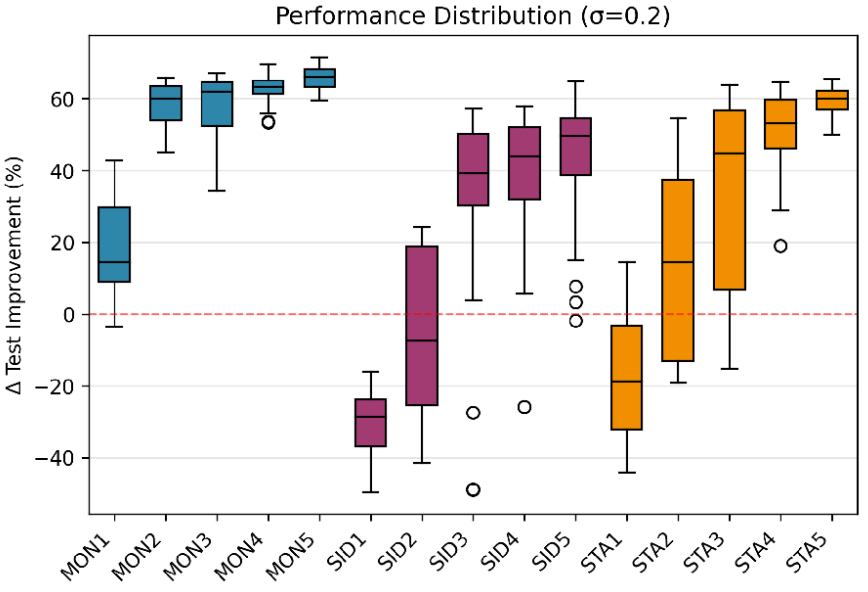
QAE training with noise

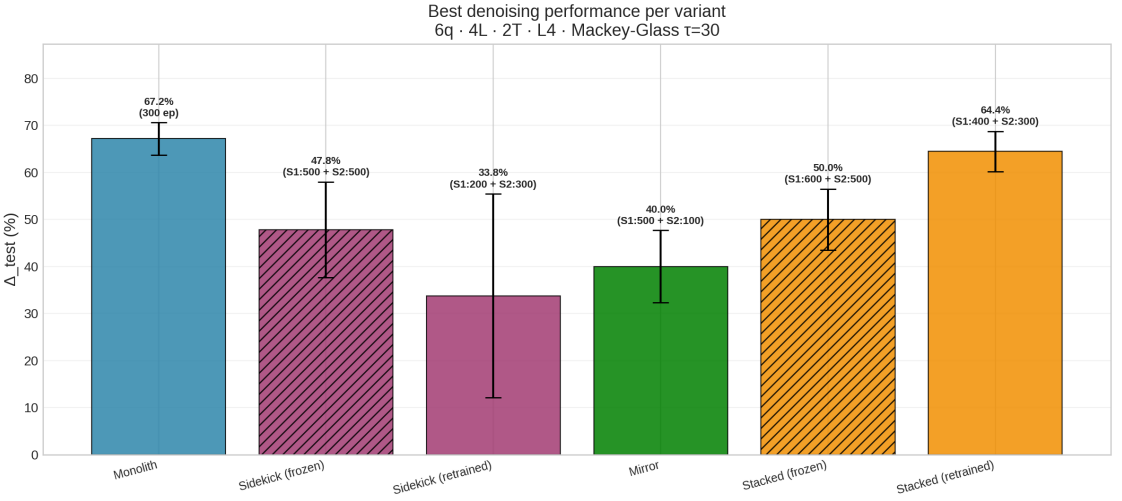
- Training QAEs capable of reducing noise from signals is challenging.
- A typical approach to such training is to do as follows:
  - Collect samples of noisy signals;
  - Collect samples of clean signals;
  - Train the system to convert noisy signals into clean signals.

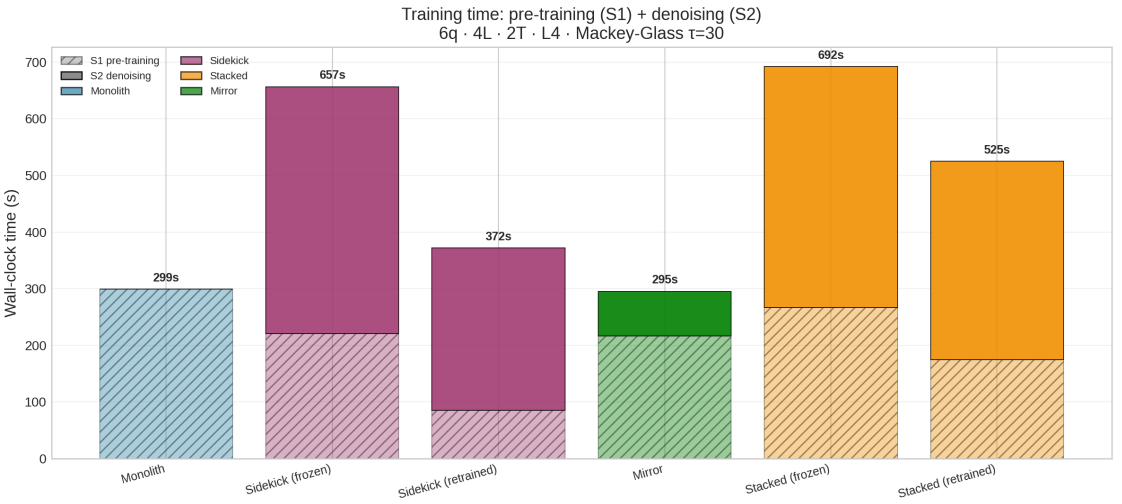


- The problem is that the samples may have been obtained at different times in different circumstances, so the system would have difficulties learning the features of clean and noisy signals, their relationships and the methods of transforming one into the other.
- A practical approach usually is to:
  - *Collect training and test samples* of clean and noisy signals;
  - Use judgement to determine *noise distribution* or its type from the collected samples;
  - *Learn how to inject noise* of this specific type into clean signals.
- Then train the model by presenting pairs of clean signals and the same signals but with noise injected.
- **Note:** It is often sufficient to inject noise of a generic distribution, e.g. uniform, Poisson, Gaussian, etc.
- **Warning:** when training a QAE model on a fixed sample of noisy and clean signals, the model will learn this noise rather than its generalisation.
- **Solution:** create new noisy samples at each training epoch. This way the model will generalise from a large number of noisy examples.

QAE architectures performance







Summary

QAE creation and training

QAE design insights

- We have discussed QAE designs that could be used in the development of models for signal and time series denoising (in PennyLane).
- We have adopted a simple, yet versatile, angle encoding of TS sliding windows. We must ensure that our measurements return identical values.
- The “monolith” full-QAE use a traditional model structure, it is simple to build and is accurate, but slower to train due to high dimensionality of its parameter space.
- The “stacked” half-QAEs use a non-conventional approach to training, their components can be trained independently with smaller number of parameters, so they are efficient to train, but may not be highly accurate.
- Hybrid quantum-classical QAEs have expanded modelling and optimisation functions, they are fast to train and highly accurate on simulators, but sub-optimal of quantum machines.
- Special training is required to deal with noise.

What quantum AEs can that classical AEs cannot

- As quantum unitaries are reversible, it is possible to train only half of the QAE to gain full-QAE functionality.
- As quantum unitaries do not lose information, it is possible to train QAEs to flow relevant information via its latent space by minimising information flowing into trash space.

Challenge tasks (see s04_challenge_qae notebook)

- Explore the Mackie-Glass chaotic time series
- Understand the PennyLane implementation of pure quantum QAE, follow the instructor’s demonstration and explanation.
- Adapt the provided notebook for execution on your computer.
- Score the pure quantum QAE on training and test partitions.
- Plot the history of training vs test scores.
- Rewrite the pure quantum QAE (in PennyLane) to the hybrid QAE model (in PennyLane + PyTorch).
- Compare pure quantum QAE vs. hybrid QAE performance.
- Improve the model performance.
- Generate 9 instances of differently initialised QAEs, and then score them and chart the results.
- Plot the data fit (example given) for the best, median and worst performing QAE.
- Reflect on this challenge.

https://github.com/ironfrown/qml_workshop_intro_v3

Thank you!

Any questions?

Photos from Unsplash
Enquanted is being somewhere in-between Enchanted and Entangled