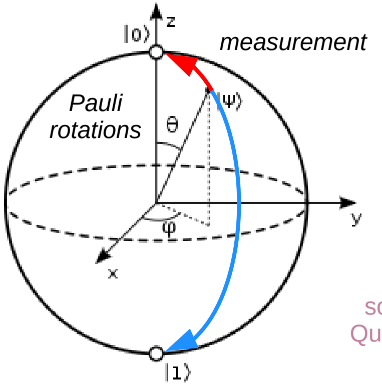
The aims of this session:

To explain methods of Quantum Time Series Analysis, its models, algorithms, benefits and challenges

Introduction to QTSA
QTSA aims, objectives and tasks
QTSA apps and algorithms
QTSA QML foundations
QTSA data encoding
QTSA state measurement
QTSA model execution
QTSA challenges
The curse of dimensionality
Expressivity vs trainability
QTSAE case study –
  Quantum TS Autoencoders
  QAE vs QML principles
Summary and conclusions

Quantum Time Series Analysis (QTSA)
+Quantum Time Series Autoencoders (QTSAE)

Jacob L. Cybulski
Enquanted, Melbourne, Australia



We will assume
some knowledge of
Quantum Computing
ML and Python

17-22 Nov 2025, JIMS, Rohini, India

Understanding the past
Predicting the future

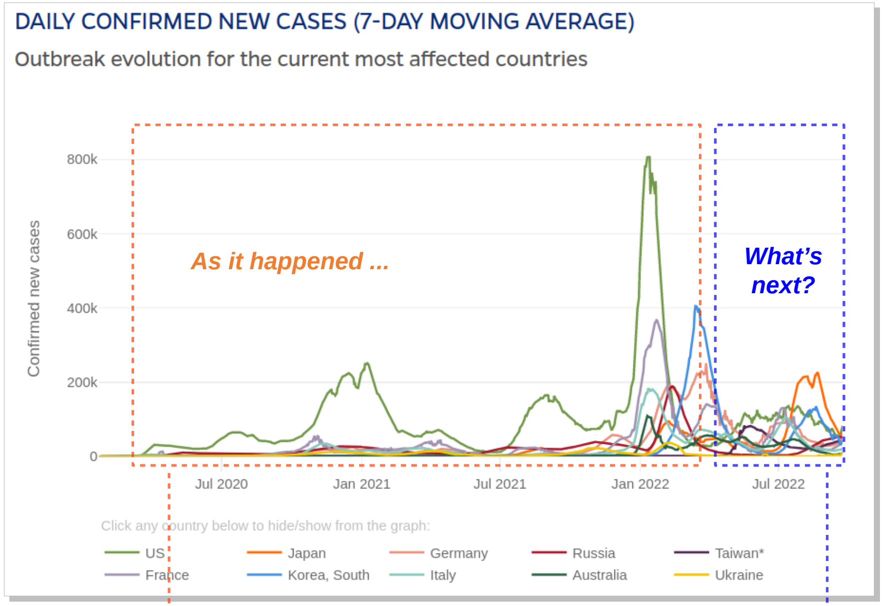
- ***Time series*** (TS) data and their visualisations inform experts, governments and the public

- Time series analysis aims to identify patterns in temporal data to support *forecasting*, *classification* and *clustering* tasks

- ***Time series analysis*** is an established discipline, with excellent tools and efficient methods, such as ARIMA or GARCH, and deep TS models (LSTM or GRU)

- Organisations that rely on time-based information are still in the pursuit of more effective analytic methods

- ***Quantum machine learning*** is a novel approach to time series analysis, with the potential for detecting *high-complexity patterns in temporal data*

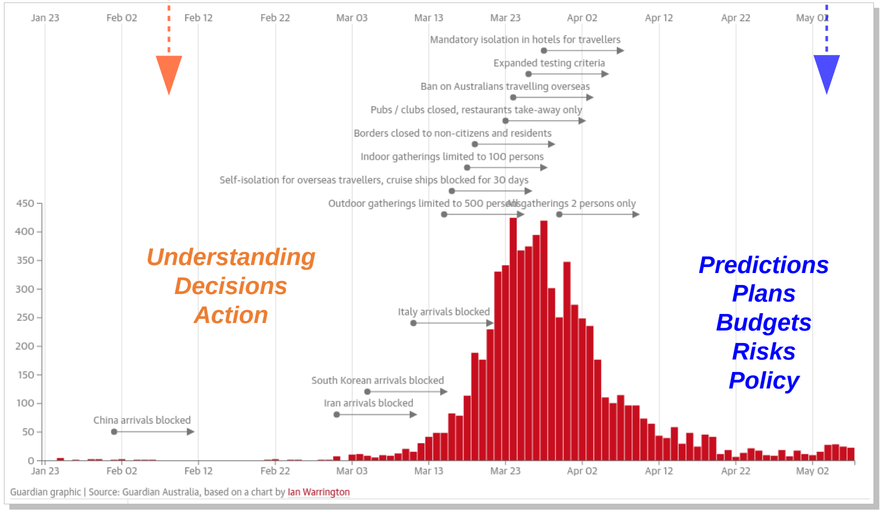


DAILY CONFIRMED NEW CASES (7-DAY MOVING AVERAGE)

Outbreak evolution for the current most affected countries

As it happened ...

What's next?



Understanding
Decisions
Action

Predictions
Plans
Budgets
Risks
Policy

Guardian graphic | Source: Guardian Australia, based on a chart by Ian Warrington

Neural Nets for Time Series Analysis

- Simple neural networks, e.g. Multi-Layer Perceptrons (MLPs), map numeric *inputs* into *outputs*, via layers of interconnected *nodes*, responsible for calculating *weighted sums* of preceding node values, which are transformed with non-linear *activation functions*

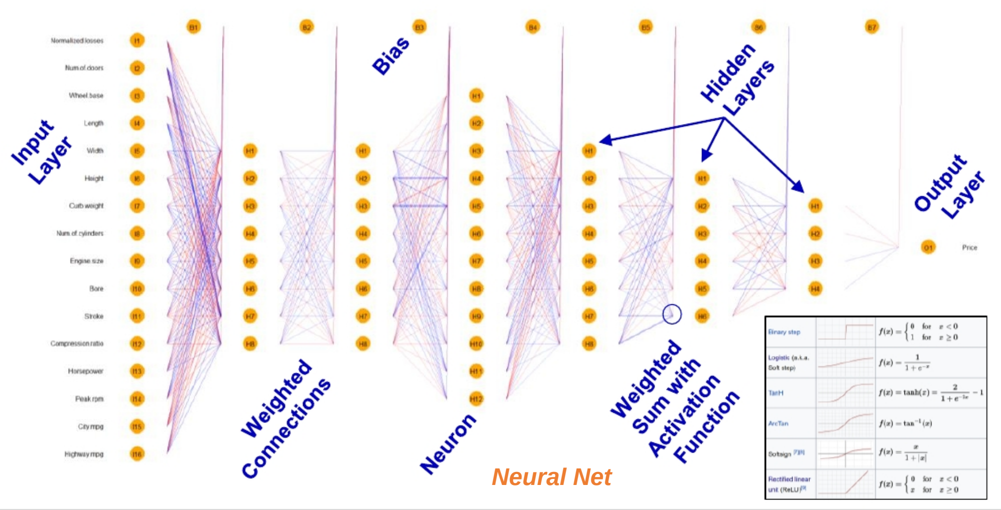
- The MLP weights are trained in the optimisation process, by matching the calculated vs expected outputs

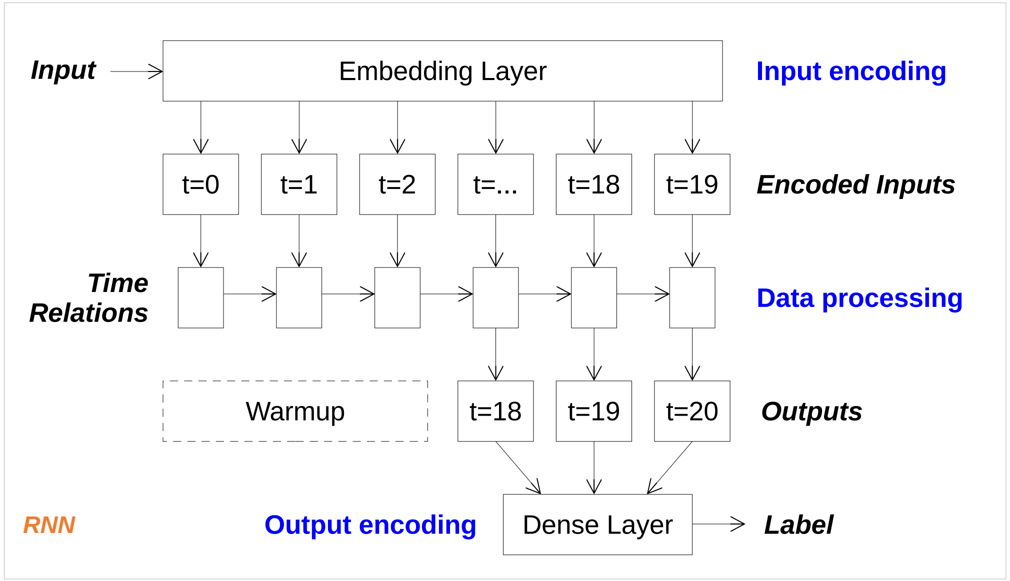
- Some neural networks can be trained for time series analysis, e.g.

  - *Recurrent Neural Networks* (RNN)
  - *Long Short-Term Memory* (LSTM) nets
  - *Gated Recurrent Units* (GRU) nets

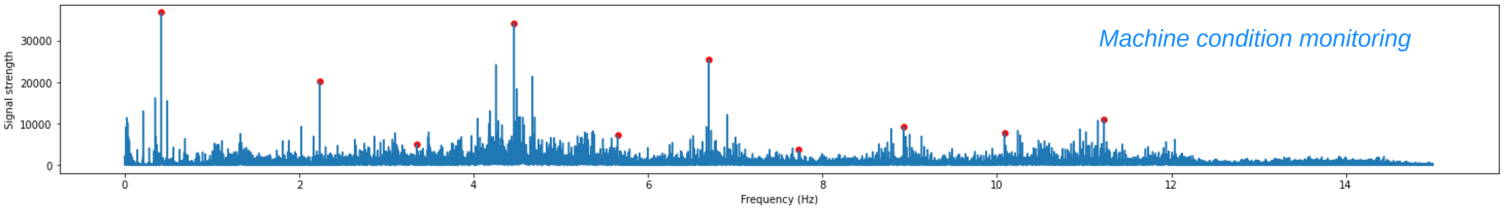
- Unlike MLPs, RNNs, LSTMs and GRUs are able to retain and rely on memory of the past data

- MLPs and RNNs are similar in their structure to some types of quantum models




Input
Embedding Layer
Input encoding
t=0 t=1 t=2 t=... t=18 t=19
Encoded Inputs
Time Relations
Data processing
Warmup
t=18 t=19 t=20
Outputs
RNN
Output encoding
Dense Layer
Label

QTSA

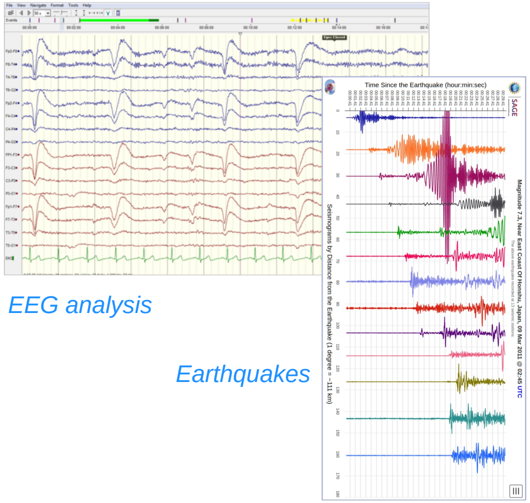


Machine condition monitoring

Quantum time series analysis (QTSA) applies Quantum Machine Learning (QML) techniques to analyse temporal data

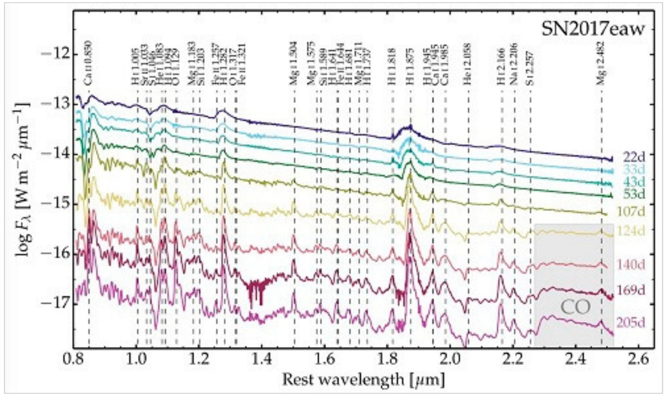
QTSA applications include medical signal monitoring, remote sensing and forecasts, machine condition monitoring, business forecasting and analytics, etc.

- TS can be *univariate* or *multivariate*
- TS display *patterns repeating over time*



EEG analysis

Earthquakes

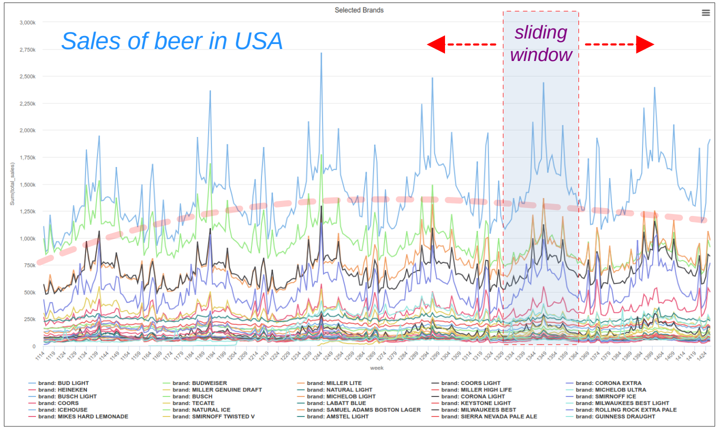


Astronomical observations

QTSA tasks

include methods of dealing with time and change, as represented in temporal data, but achieved with QML algorithms, e.g.

- by setting the task in terms of *curve-fitting* of temporal data to some function (below left)
- by presenting a problem as *forecasting* describing changes in terms of pattern-matching between fragments of the past and the future (below right).



Sales of beer in USA

sliding window

all data are temporal

all suffer from noise and anomalies

all could potentially lead to catastrophic failures

Acknowledgements:

Gearbox and Vibration Analysis ML, 2023, Nakkeeran, Kaggle.com

Gemini/GNIRS spectra, 2017, NOIRLab, Wikimedia.

Bronnenberg, B.J., Kruger, M.W., Mela, C.F., 2008. Database Paper —The IRI Marketing Data Set. Marketing Science 27, 745–748. https://doi.org/10.1287/mksc.1080.0450

Earthquake, Mag 7.3, East Coast of Honshu, Japan, 2011, The Global Seismogram Viewer, http://ds.iris.edu/gsv

EEG of brain and heart action, 2012, Otoomuch, Wikimedia.

Apps, devs and issues

Sample QTSA applications (input → output)

Apps are found in Sci & Eng, Earth Sci, Finance, Meds, etc.

- Explanation (sequence → function)
- Decision support (sequence → logical)
- Function fitting and forecasting (points/sequence → points)
- Monte Carlo + Random walks (constraints → sequence)
- Noise and anomaly elimination (sequence → sequence)

Difficulties in processing TS data

- TS are of high volume
- TS have tacit features
- Neighbouring TS features are highly dependent
- Consecutive TS values are homogenous
- TS have non-local patterns, e.g. cyclicity / seasonality
- TS suffer from noise, anomalies and volatility
- TS data ages quickly
- Statistical / classical approaches to TS analysis require very strict pre-processing of time-based data

Some QML algorithms that deal with time and change

Time in data

- Quantum Sequence Models (QRNN, QLSTM, QGRU)
- Quantum Reservoir Computing (QRC)
- Quantum Self-Attention and Transformers (LLMs)
- Quantum Fourier Analysis (QFT, PQFT, QFFT)

Change and state evolution

- Quantum Optimisation Algorithms (QAOA, QUBO)
- Quantum Annealing / Quantum Adiabatic Algorithm (QAA)
- Quantum Reinforcement Learning (QRL)
- Quantum Bayesian Modelling (QBN, QBC, QBNN)
- Quantum Genetic Algorithms (QGA)

Supporting models

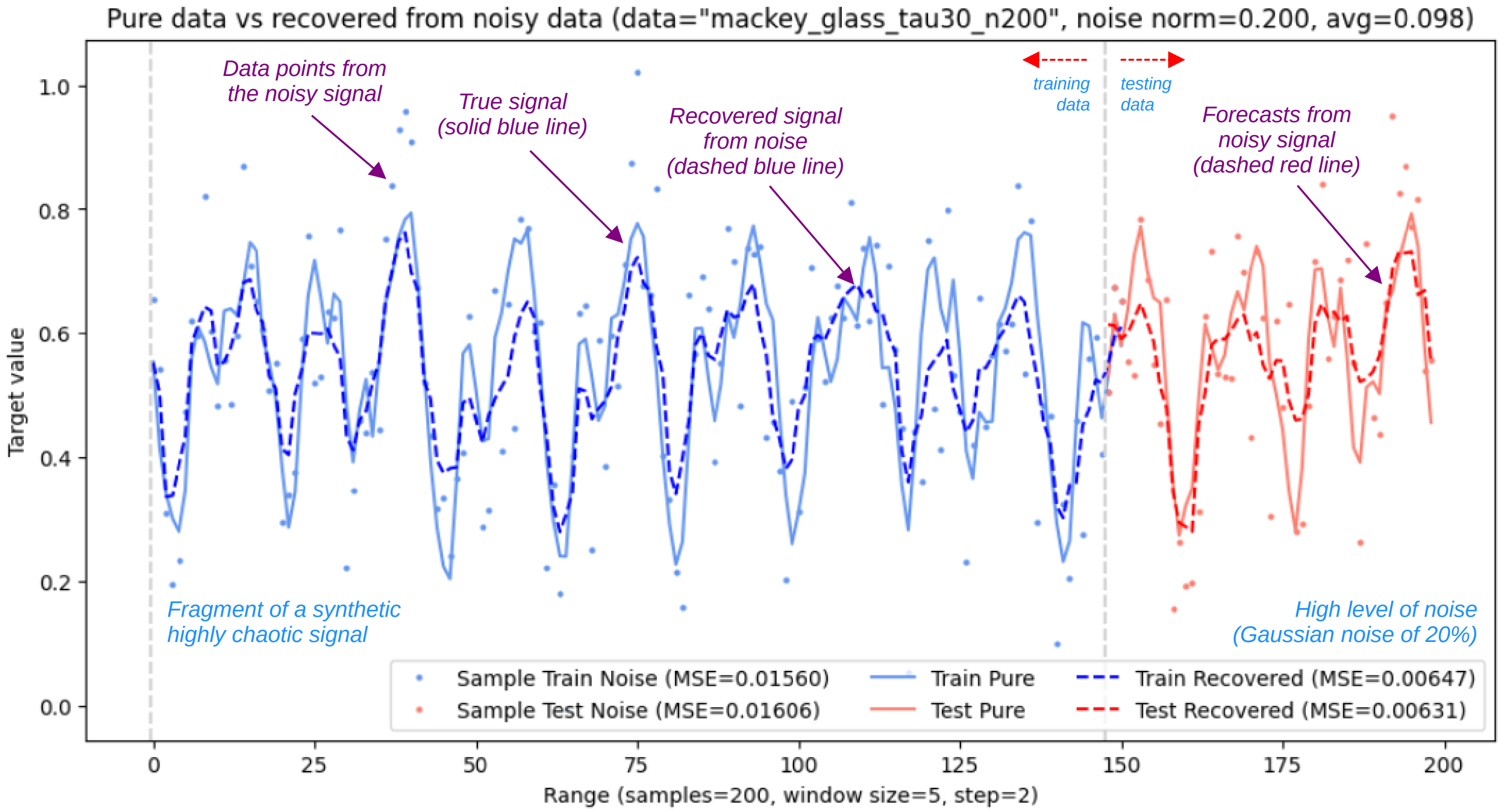
- Quantum Neural Networks (QNN, VQC/R, QCNN, qGAN)
- Quantum Support Vector Machines (QSVM, QSVC/R)
- Quantum Kernel Methods (Feature Maps, Estimators)
- Quantum Clustering Algorithms (QCA k-NN, DQC)

Olivier Ezratty, Understanding Quantum Technologies, 7th Ed (2024)

Maria Schuld and Francesco Petruccione.
Machine Learning with Quantum Computers. 2nd ed. Springer, 2021.

Sample problem: *reduce noise from audio recordings*
Possible solution: *apply quantum autoencoders*

What is needed to create a quantum model capable of representing and processing of temporal data?



Data points from the noisy signal

True signal (solid blue line)

Recovered signal from noise (dashed blue line)

Forecasts from noisy signal (dashed red line)

Fragment of a synthetic highly chaotic signal

High level of noise (Gaussian noise of 20%)

Questions:
Should we simply average signals?
Should we apply a model-based denoising?

Other questions:
What is normal?
What is abnormal?

Answers:
We can develop a QTSA model to reduce noise from signals!

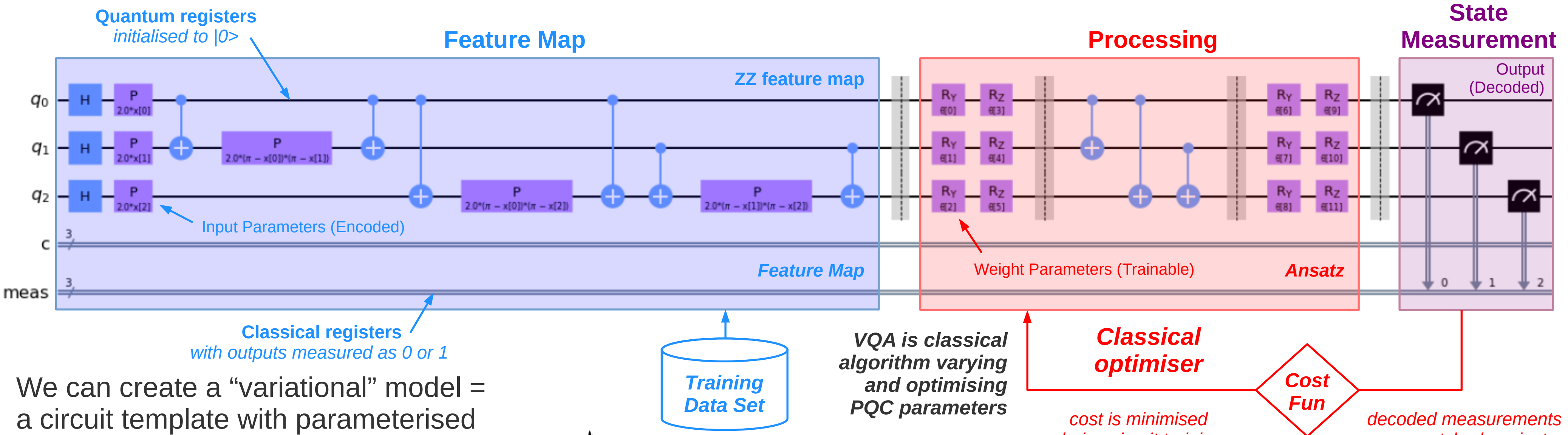
Extensions:
We can then use a noise-free signal to identify signal anomalies!

TS QAE implementation in PennyLane or Qiskit

QML Principles:

Parameterised Quantum Circuits (PQC) and Variational Quantum Algorithms (VQA)

PQCs are not executable!
They must first be instantiated, i.e. all of their
input and weight parameters must be assigned values!



Quantum registers
initialised to |0>

Feature Map

Processing

State Measurement

ZZ feature map

Input Parameters (Encoded)

Feature Map

Output (Decoded)

Weight Parameters (Trainable)

Ansatz

Classical registers
with outputs measured as 0 or 1

Training Data Set

VQA is classical algorithm varying and optimising PQC parameters

Classical optimiser

Cost Fun

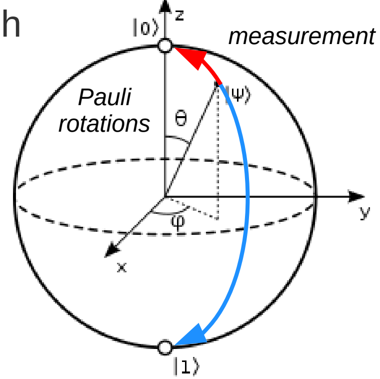
cost is minimised during circuit training

decoded measurements are matched against training data

We can create a "variational" model = a circuit template with parameterised gates, e.g. P(a), Ry(a) or Rz(a), each allowing rotation of a qubit state in x, y or z axis (as per Bloch sphere).

Typically (but now always), such circuits consist of three blocks:

- a feature map (TS input)
- an ansatz (TS processing)
- measurements (TS output)



measurement

Pauli rotations

Classical input data is encoded (embedded) into the feature map's parameters, setting the model's initial quantum state.

The quantum state is altered by an ansatz, of parameterised quantum gates, which are trained by a classic optimiser

The circuit final state is measured and decoded (interpreted) as the model's output in the form of classical data.

QML Principles:
Ansatz design

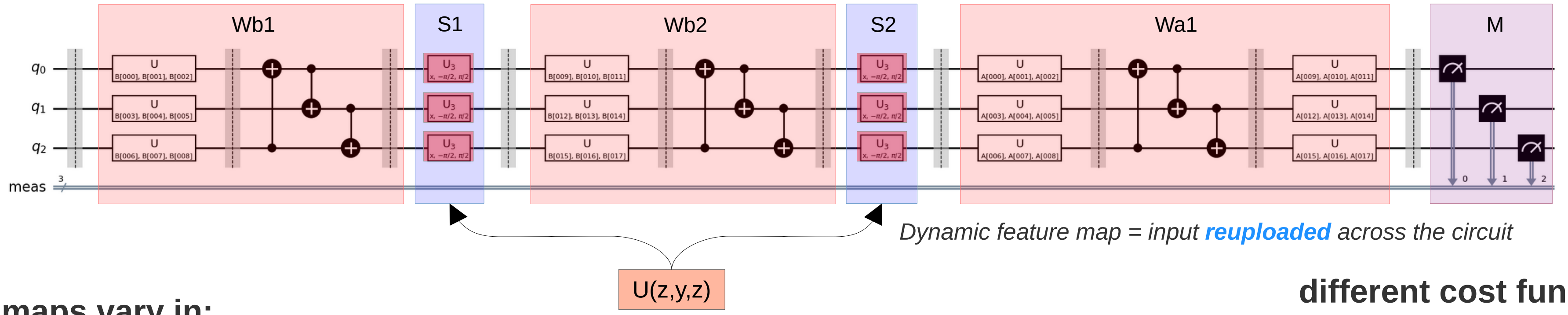
Beware that (sometimes)
adding 1 more measurement
doubles the number of outcomes!

The number of outcomes
grows exponentially with the
number of measurements!

Beware that
adding 1 extra qubit adds
parameters and entanglements!

The number of circuit states
grows exponentially with the
number of qubits!

Encoding of classical data in a quantum circuit is
***not** what our ML experience tells us about **inputs** !*



*Dynamic feature map = input **reuploaded** across the circuit*

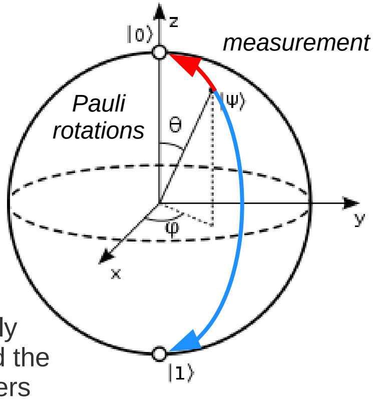
feature maps vary in:
structure and function

ansatze vary in:

- width (qubits #)
- depth (layers #)
- dimensions (param #)
- structure (e.g. funnelling)
- entangling (circular, linear, sca)

ansatz layers consist of:
rotation blocks and entangling blocks
of U(z, y, z) and CNOT gates
(rotations) (entanglement)

rotation gates
alter qubit states
around x, y, z
axes



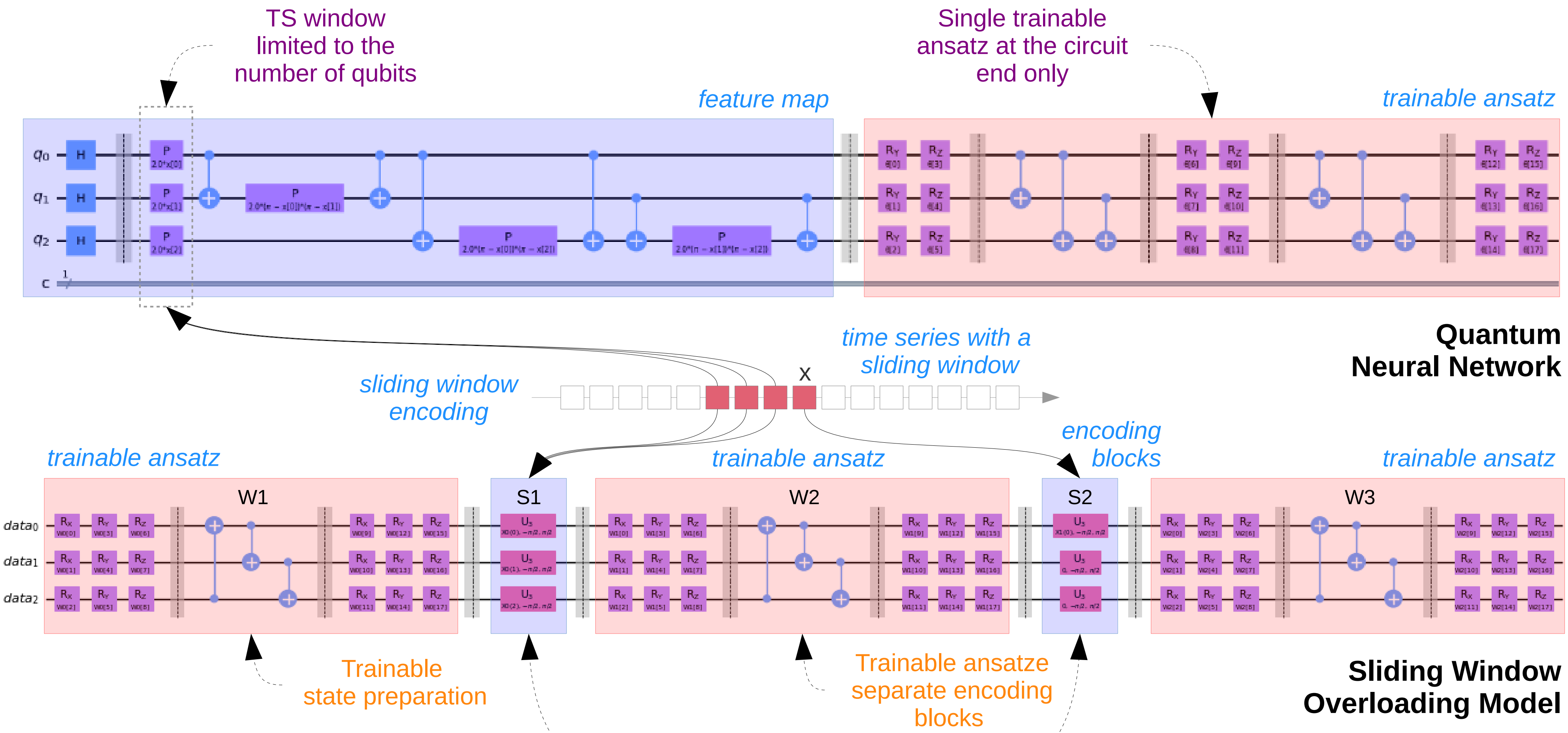
To execute a
circuit we just apply
it to input data and the
optimum parameters

different cost functions:
R2, MAE, MSE, Huber, Poisson, cross-entropy,
hinge-embedding, Kullback-Leibner divergence

different optimisers:
gradient based (Adam, NAdam and SPSA)
linear approximation methods (COBYLA)
non-linear approximation methods (BFGS)
quantum natural gradient optimiser (QNG)

circuit execution on:
simulators (CPUs), accelerators (GPUs) and
real quantum machines (QPUs)

Jacob L. Cybulski and Sebastian Zając (2024): "The Art of Data Encoding and Decoding for Quantum Time Series Analysis." 8th International
Conference on Quantum Techniques in Machine Learning University of Melbourne, Melbourne, Australia, 25-29 November 2024.



TS window limited to the number of qubits

Single trainable ansatz at the circuit end only

Quantum Neural Network

Trainable state preparation

Trainable ansatze separate encoding blocks

Sliding Window Overloading Model

Qubit **overloading** / Unlimited size of TS window

Potential for encoding multivariate TS

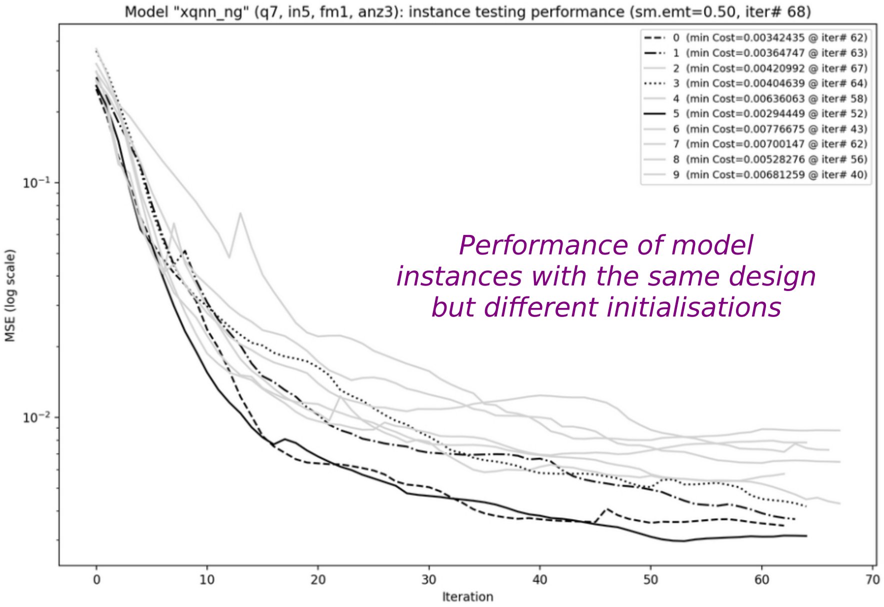
QTSA

Does QTSA design matter?

Each approach to QTSA model’s data encoding, their processing, and state measurement enhance or impede:

model expressivity, i.e. its ability to effectively represent time series data in quantum (Hilbert) space; as well as,

model trainability, i.e. its capacity to learn and generalise for predictive accuracy and efficiency in the process of model optimisation.



Performance of model instances with the same design but different initialisations

Are QTSA models better than classical NN models?

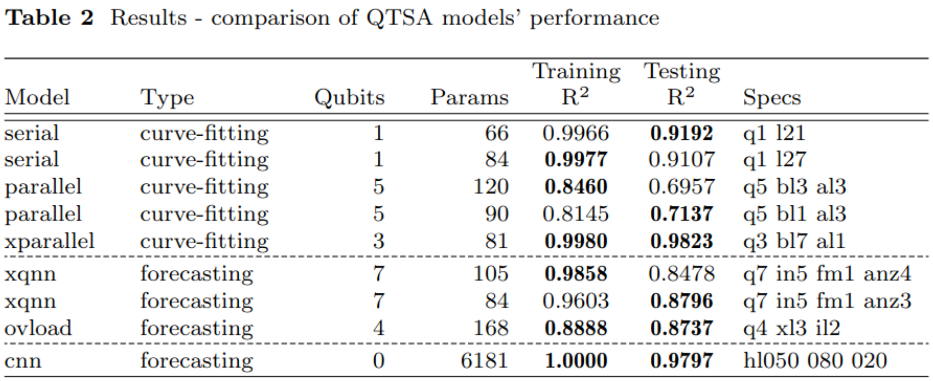


Table 2 Results - comparison of QTSA models’ performance

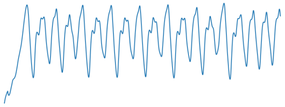
Model	Type	Qubits	Params	Training R^2	Testing R^2	Specs
serial	curve-fitting	1	66	0.9966	**0.9192**	q1 l21
serial	curve-fitting	1	84	**0.9977**	0.9107	q1 l27
parallel	curve-fitting	5	120	**0.8460**	0.6957	q5 bl3 al3
parallel	curve-fitting	5	90	0.8145	**0.7137**	q5 bl1 al3
xparallel	curve-fitting	3	81	**0.9980**	**0.9823**	q3 bl7 al1
xqnn	forecasting	7	105	**0.9858**	0.8478	q7 in5 fm1 anz4
xqnn	forecasting	7	84	0.9603	**0.8796**	q7 in5 fm1 anz3
ovload	forecasting	4	168	**0.8888**	**0.8737**	q4 xl3 il2
cnn	forecasting	0	6181	**1.0000**	**0.9797**	hl050 080 020

Different designs and median performance across differently initialised model instances

Jacob L. Cybulski and Sebastian Zając (2025):
"Quantum Modelling of Time Series: Expressivity vs. Trainability." 20 August 2025,
Preprint (V1), Research Square [https://doi.org/10.21203/rs.3.rs-7253212/v1].

Quantum state evolution

Metaphor for quantum state evolution



State Preparation

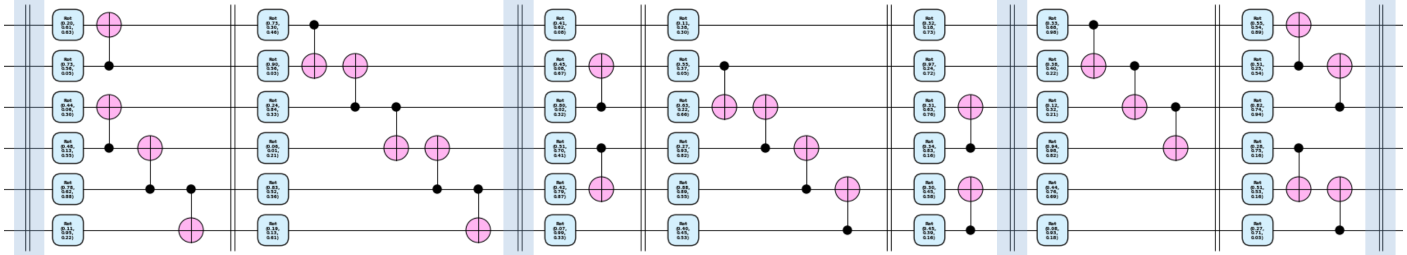
Classical Data → Quantum State

matters

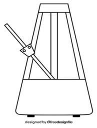
*Initial (data) state →
Problem state space*

*Problem state space →
Solution state space*

*Solution state space →
Results state space*



matters



State Measurement

Quantum State → Classical Data

matters

Advantages and **Challenges** in QTSA model development:

- ***High-dimensional parameter/state space*** (large circuits)
☑ enables linear separation of quantum information, but
☒ flattens the gradient space.

- ***Highly entangled circuits*** (complex circuits)
☑ inter-relate qubits and their training parameters, but
☒ suffer from non-local gradients, complex state dynamics, cost landscape complexity, and high decoherence rate.

It matters what quantum architecture is to be used!

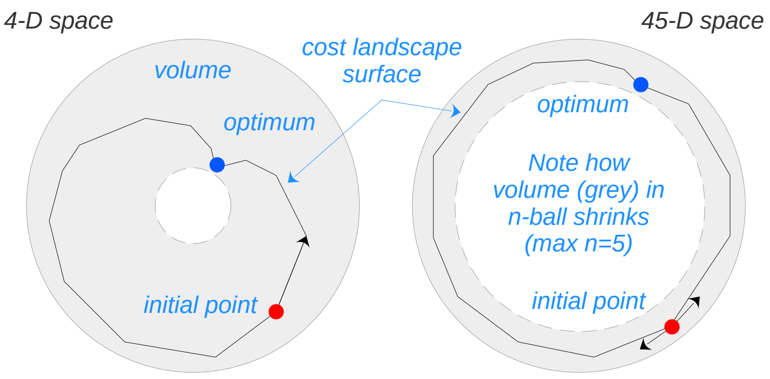
- ***Global cost*** (measuring all qubits)
☑ allows better utilisation of model training parameters, but
☒ requires exponential increase of the circuit runs, needed to prevent sparse distribution of outcomes.

- ***Model initialisation*** (optimisation starting point)
☑ quick, easy and popular random params initialisation, but
☒ proven to make model optimisation ineffective.

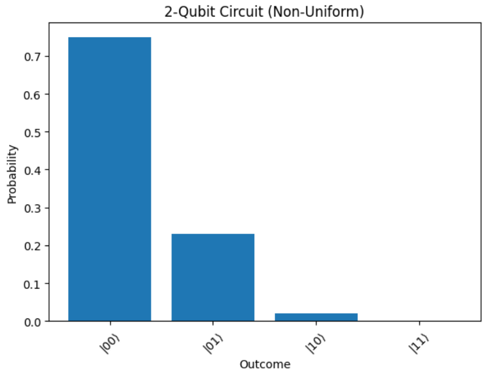
All of the above are known to attract *barren plateaus*!

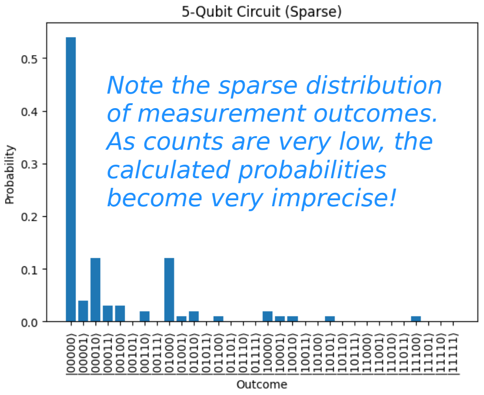
Cybulski, J.L., Nguyen, T., 2023. “Impact of barren plateaus countermeasures on the quantum neural network capacity to learn”, Quantum Inf Processing 22, 442.

The curse of *dimensionality*



Note how volume (grey) in n-ball shrinks (max n=5)





Note the sparse distribution of measurement outcomes. As counts are very low, the calculated probabilities become very imprecise!

Barren Plateaus

(too many dimensions = qubits and/or parameters)

- Pairwise distances between uniformly distributed points in high-dimensional space become (almost) identical, and its surface is (almost) flat (fig. left).
- In a quantum model with a high-dimensional parameter space, the cost landscape also becomes (almost) flat, called ***barren plateau (BP)***.
- When BPs emerge, the optimiser struggles finding the optimum model parameters.
- Selecting the optimisation initial point far from the optimum (e.g. random) makes it even more difficult!

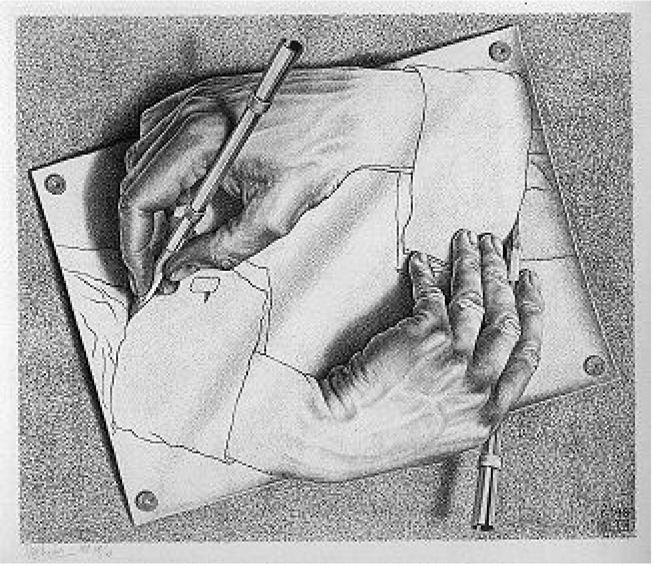
Sparse measurement outcomes

(too many measurements, esp. as probability distribution)

- When increasing the number of measurements, we also exponentially increase the number of outcomes, leading to the exponential increase in the number of circuit runs!
- Unless the number of runs is increased with measurements, distribution of outcomes becomes sparse and the probability calculations become imprecise (fig. left).

QTSA case study

Quantum TS Autoencoders



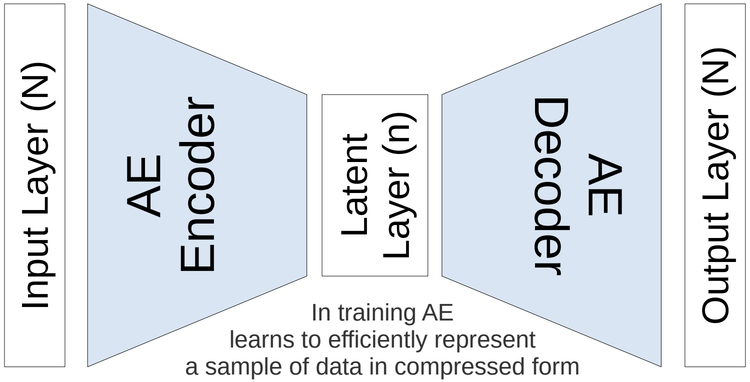
Fair use, Wikipedia:
https://en.wikipedia.org/w/index.php?curid=3475111

QML principles used in QTSAE modelling:

- *Unitary quantum operations* perform norm-preserving linear transformation of quantum states in quantum (Hilbert) space.
- *All standard quantum gate operations are unitary*, e.g. state rotations (Rx, Ry, Rz, X, H) and entanglements (CNOT).
- *Blocks, circuits and combinations of unitary operations are also unitary*.
- *Unitary operations preserve quantum information*, which means that in response to their action, quantum information shifts between qubits and their gates but is never lost.
- *Unitary models are reversible*, i.e. applying all operations in reverse order (perform adjoint) fully reverses their effect on quantum state.
- *Unitary operations can be treated as differentiable functions*, so their behaviour can be analysed using standard mathematical approaches, gradient-based optimisers rely on differentiability of quantum circuits.
- *Some quantum operations destroy quantum state and are not unitary*, e.g. qubit initialisation, mid-circuit resetting, and state measurement, and in the process we lose significant parts of quantum information.

Classical Autoencoders & Quantum Autoencoders

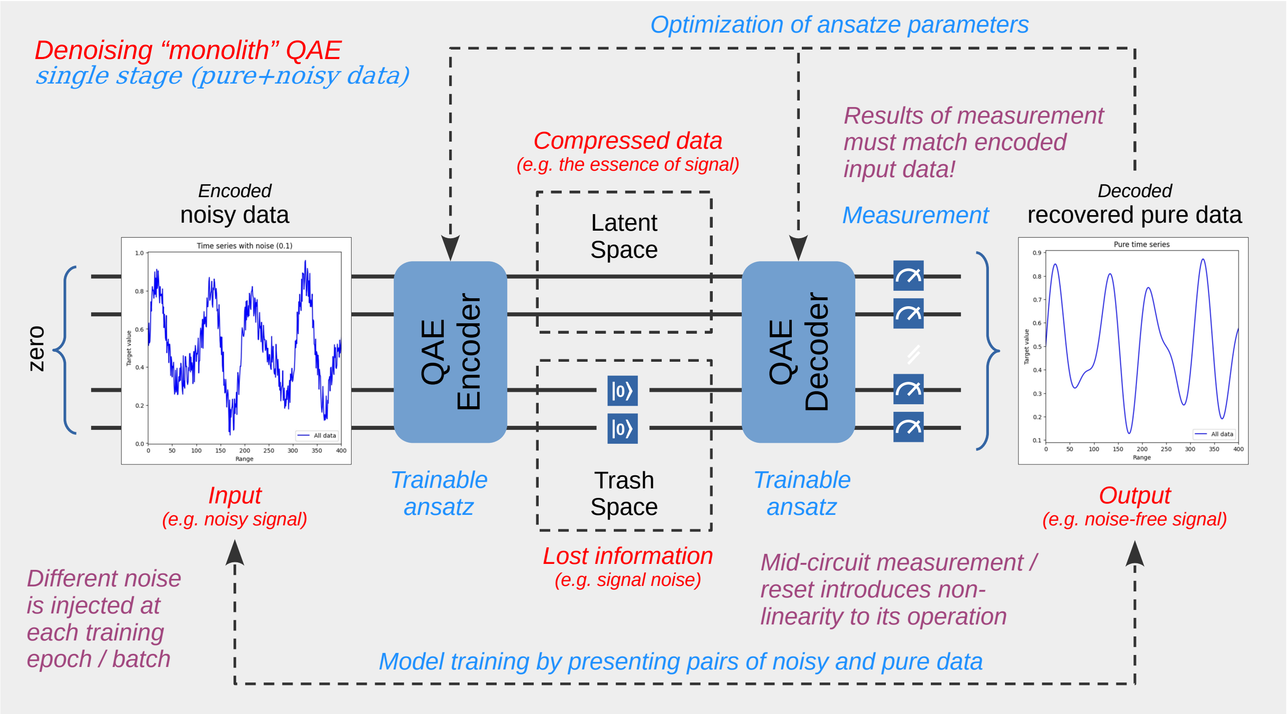
- ***Autoencoders (AE)*** are ML models that encode information into a compact / compressed form, from which approximate information content can be decoded effectively
- In the process AEs lose the infrequent, insignificant or unwanted parts of information
- Typically, an AE is implemented as a multilayer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



In training AE
learns to efficiently represent
a sample of data in compressed form

- ***Quantum Autoencoders (QAE)*** utilise QML to implement AE
- There are still few practical applications of QAEs
- QAEs have the potential to assist in detection, removal or reduction of highly complex noise and anomaly patterns
- We will apply QAE to time series analysis
- Some aspects of QAEs design **cannot** be achieved with classical ML!

The “monolith” architecture
of TS denoising QAE



Optimization of ansatze parameters

Denoising “monolith” QAE
single stage (pure+noisy data)

Compressed data
(e.g. the essence of signal)

Results of measurement must match encoded input data!

Encoded
noisy data

Latent Space

Measurement

Decoded
recovered pure data

zero

QAE Encoder

QAE Decoder

|0⟩

|0⟩

Input
(e.g. noisy signal)

Trainable ansatz

Trash Space

Trainable ansatz

Output
(e.g. noise-free signal)

Lost information
(e.g. signal noise)

Different noise is injected at each training epoch / batch

Mid-circuit measurement / reset introduces non-linearity to its operation

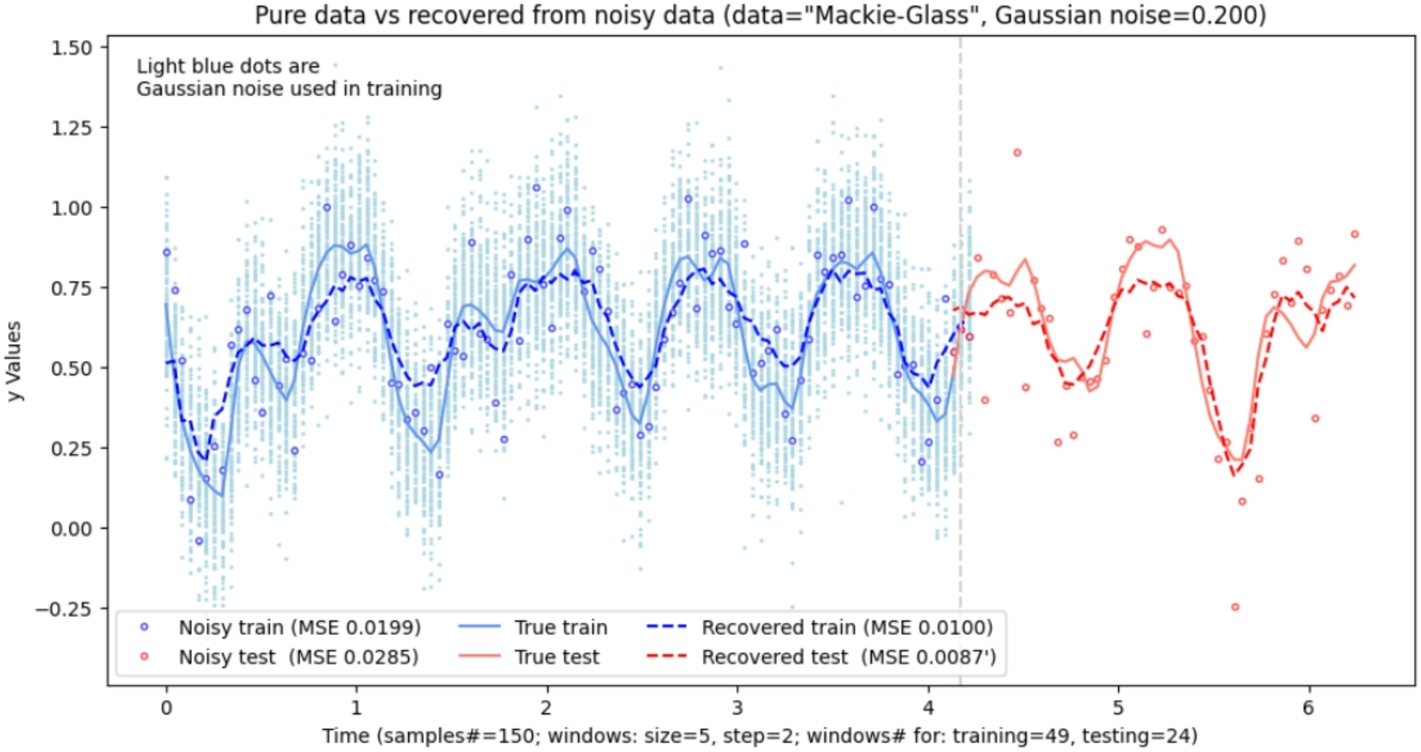
Model training by presenting pairs of noisy and pure data

The “monolith” QAEs potentially have many weights, and so their training can be computationally expensive. Therefore, we are looking for alternative designs!

QAE:
Restating AE principles in QML terms

- *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

Training a denoising “monolith” QTSAE

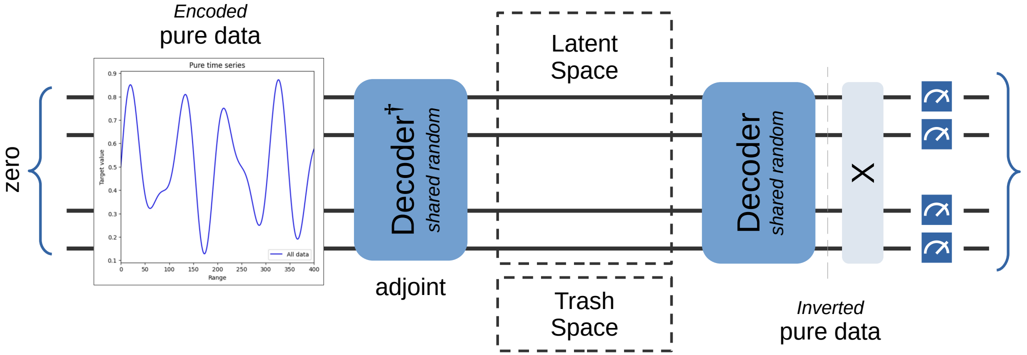


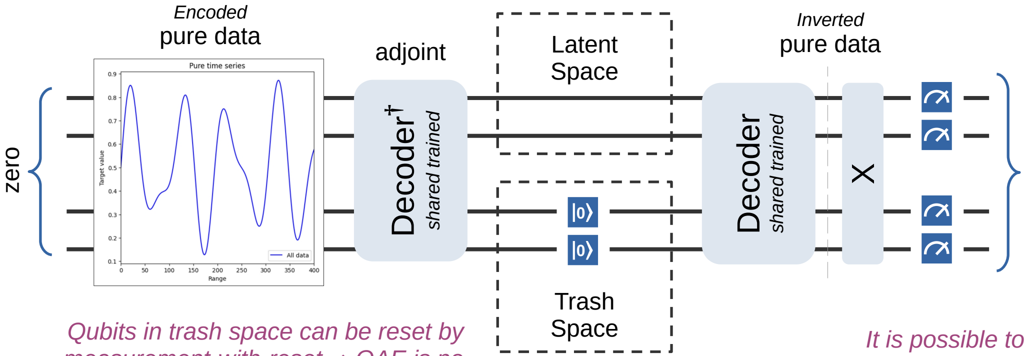
- Training denoising QTSAE commonly involves injecting synthetic noise into clean TS data at each optimisation cycle
- Then we train the model by presenting pairs of clean and noisy TS signals
- It is often sufficient to inject noise of a generic distribution, e.g. Uniform, Poisson, Gaussian, etc.

Jacob L. Cybulski and Sebastian Zając (2024): "Design Considerations for Denoising Quantum Time-Series Autoencoder."
In Proc. of 24th International Conference on Computational Science (ICCS), edited by Leonardo Franco, Clélia de
Mulatier, et al, Lecture Notes in Computer Science, LNCS, vol. 14837, Part VI, 252–67, July 2-4, 2024, Malaga, Spain.

Replicating QAE with half-QAE

Let's design a QAE to consist of encoder and decoder unitaries to have mirror image structures, i.e. a QAE Encoder is an adjoint of a QAE Decoder!





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

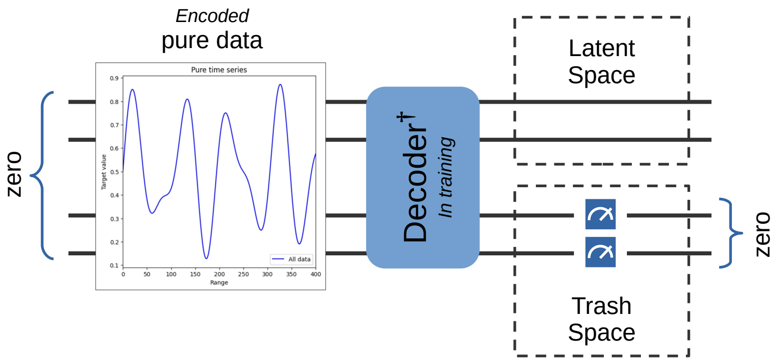
When the latent space spans all qubits, QAE is unitary, therefore there is **no loss of information**.

As the QAE Encoder is an **inverse** of the QAE Decoder, hence they always cancel each other operations – regardless of their weights.

When we reset trash space, QAE is no longer unitary and we will be **destroying information** flowing from the encoder to decoder.

However, we can train the QAE Encoder / Decoder to **reduce 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**.

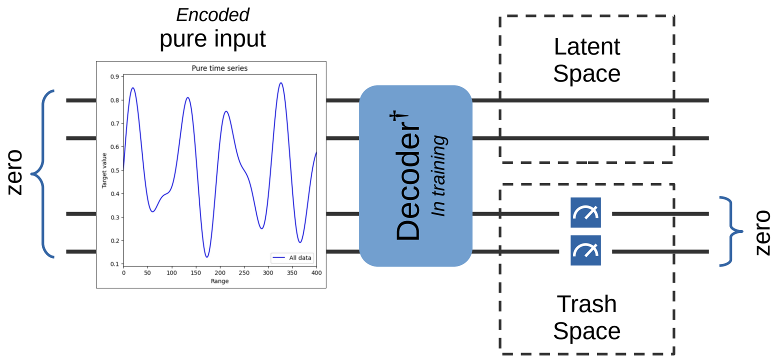


Denoising TS with Stacked half-QAEs

When we introduce noisy signal on input, the replicating QAE are able to reduce signal noise by adjusting the trash size and weights.

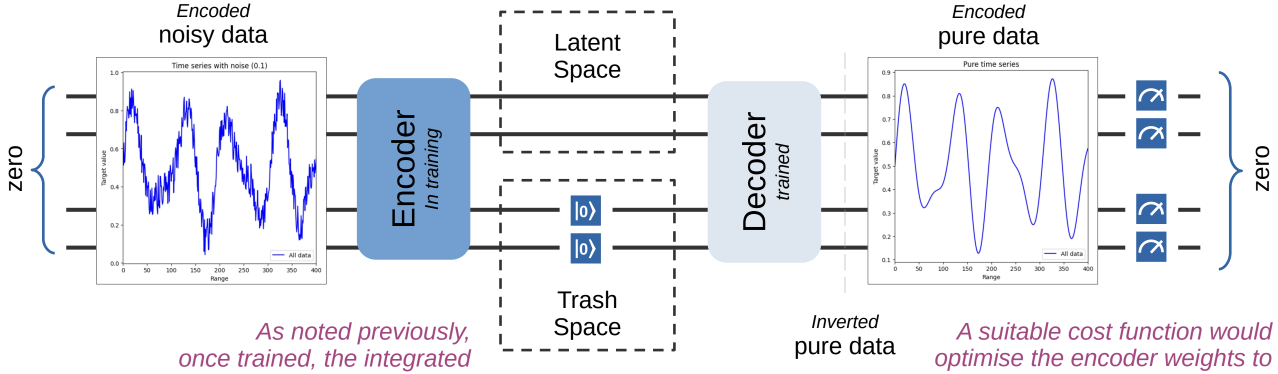
However, **the replicating QAEs are limited!**

QAE can be further improved by breaking their weight symmetry and training them with noisy and pure signals in two phases.



Phase 1: As before, we train an inverted QAE decoder using pure data and a cost function aiming to converge trash to zero.

This cannot be done using classical methods!



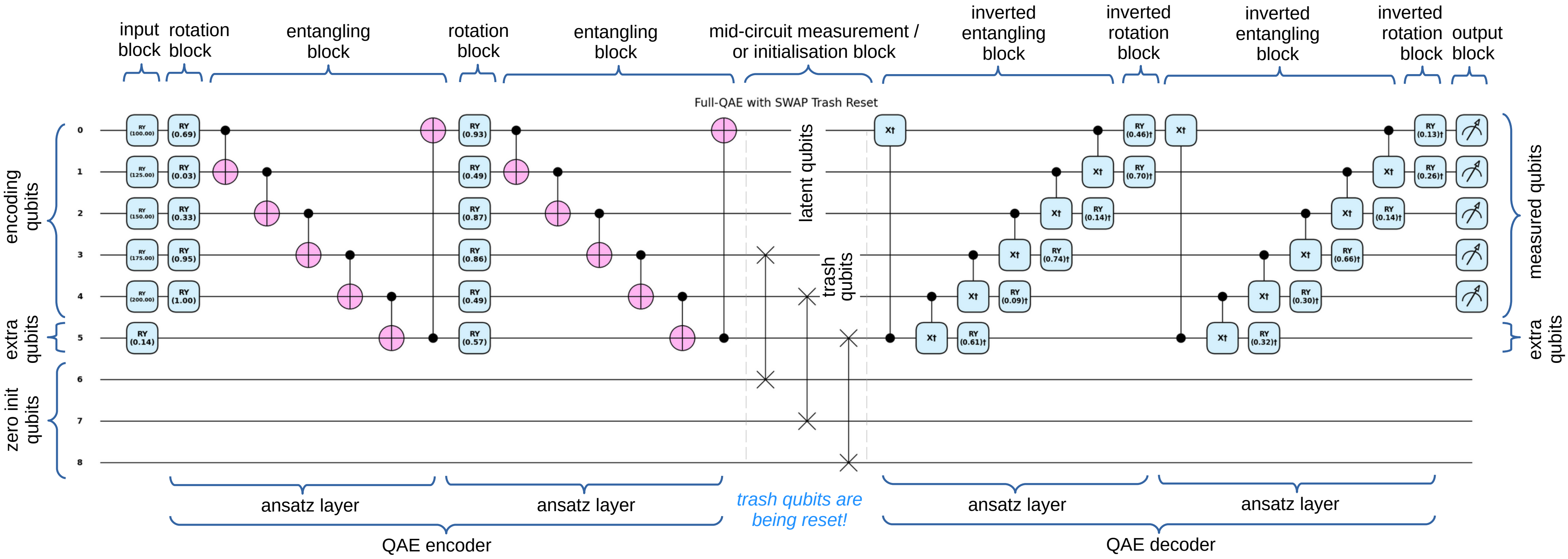
As noted previously, once trained, the integrated circuit will require to invert its output with X on every qubit

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

Phase 2: We then train a QAE encoder with noisy data, in combination with an inverted and previously trained QAE decoder, which will produce an inverse of pure data approximation.

In a perfect QAE this output would cancel pure data encoding, so we aim to converge it to zero.

Anatomy of a unitary QAE



input block

rotation block

entangling block

rotation block

entangling block

mid-circuit measurement / or initialisation block

inverted entangling block

inverted rotation block

inverted entangling block

inverted rotation block

output block

encoding qubits

extra qubits

zero init qubits

latent qubits

trash qubits

measured qubits

extra qubits

ansatz layer

ansatz layer

trash qubits are being reset!

ansatz layer

ansatz layer

QAE encoder

QAE decoder

QAEs resetting trash qubits mid-circuit are no longer unitary, hence not differentiable, and slow to optimise.

Instead we can apply SWAP operations with zero initialised qubits, as shown here.

Such circuits are unitary, and so differentiable, and can be optimised using gradient-based methods (e.g. ADAM).

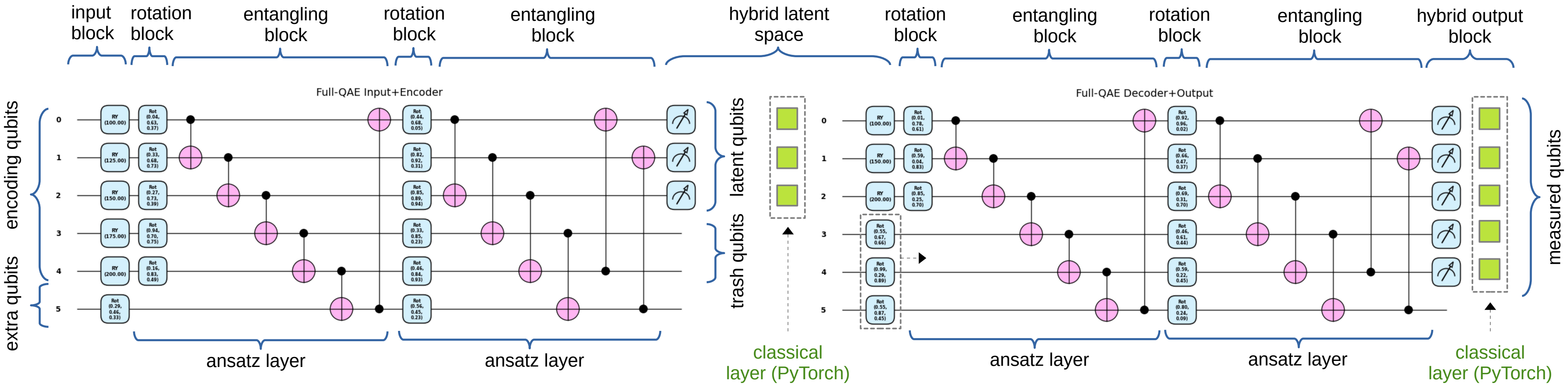
Note that QAE nsatze may be extended with extra qubits to create more trainable parameters.

Anatomy of Hybrid QAEs

Hybrid QAE encoder and decoder

As unitary circuits are differentiable, therefore, they can be combined into systems composed of quantum and classical components. For example, it is possible to form a hybrid QAE of two quantum circuits separated by layers of classical neural networks. Such models can be trained very efficiently!

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.

Summary

QTSA insights

- The aims of quantum time series analysis (QTSA) is to apply principles of Quantum Machine Learning to the analysis of temporal data
- QTSA applications include medical signal monitoring, remote sensing and forecasts, machine condition monitoring, business forecasting and analytics, etc
- QTSA considers time as an external variable presiding over changes in observable phenomena, which are represented in and acted on by quantum models and algorithms
- QTSA needs to deal with common challenges of temporal data, such as its high volume, tacit features, feature dependency, homogeneity of consecutive values, non-local patterns, noise, anomalies and volatility, short life-time
- QML offers many quantum models and algorithms of use to QTSA tasks

QTSAE insights

- QTSA algorithms rely on parameterised quantum circuits and hybrid quantum-classical variational quantum algorithms
- QTSA models typically consist of a feature map encoding data, an ansatz of parameterised gates and measurements
- QTSA TS encoding is challenging, due to limited qubit resources, so it benefits from data reuploading / overloading
- QTSA development challenges include dealing with high-dimensional parameter space, highly entangled circuits, global measurements and parameter initialisation strategies
- QTSA high-dimensionality is its curse!
- QTSA success = adopting a suitable model architecture, which could deliver its high expressivity as well as its trainability
- TS QAEs are QTSA models, which take advantage of QML features absent from classical ML models, such as preservation of quantum information, differentiability and reversibility of model unitaries!

Thank you!

Any questions?

Photos from Unsplash and Wikipedia

Enquanted is being somewhere in-between Enchanted and Entangled