The aims of this session:

To explore and understand encoding strategies for credit risk data

- *Klasyczne sieci neuronowe*
- *Obliczenia kwantowe*
- *Quantum Neural Networks*
- *Kodowanie klasycznych danych*
- *Pomiar i jego interpretacja*
- *Trenowanie QNN*
- *QNN vs klasyczne NN*
- *Wykrywanie anomalii w ryzyku kredytowym*
- *Podsumowanie*

Kwantowe sieci neuronowe jako modele ryzyka kredytowego

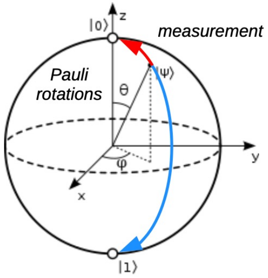
Teoretyczne i praktyczne aspekty kwantowych sieci neuronowych

Sebastian Zając

SGH Szkoła Główna Handlowa w Warszawie

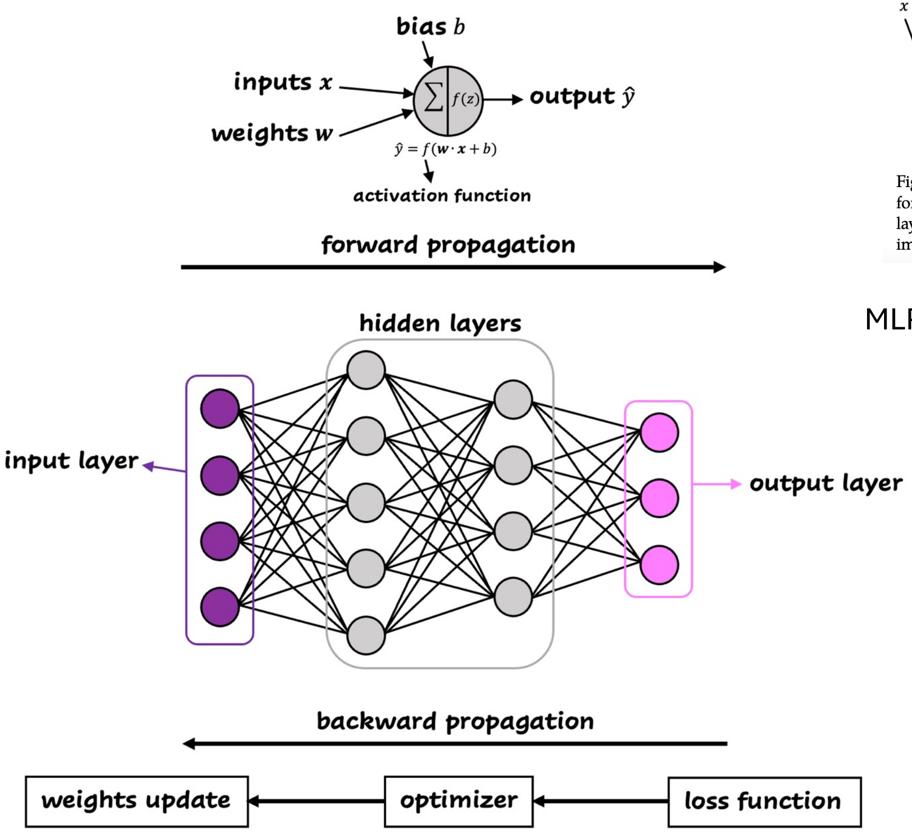
Jacob L. Cybulski

Enquanted, Melbourne, Australia



07-08.04.2025, Modelowanie Preferencji a Ryzyko

Neural Networks



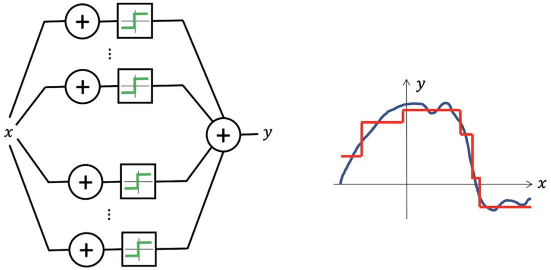


Figure 1: Multilayer Perceptrons (Rosenblatt, 1958), the simplest feed-forward neural networks, are universal approximators: with just one hidden layer, they can represent combinations of step functions, allowing to approximate any continuous function with arbitrary precision.

MLP = *Universal Approximation Theorem*

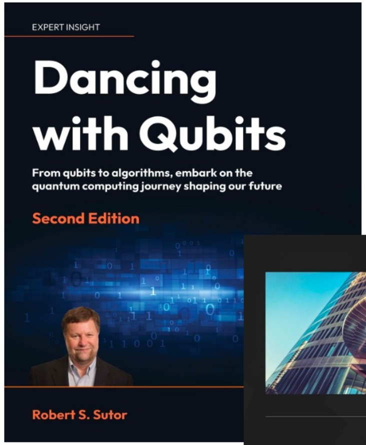
$$y = \kappa(z) = \kappa\left(\sum_i w_i x_i + b\right)$$

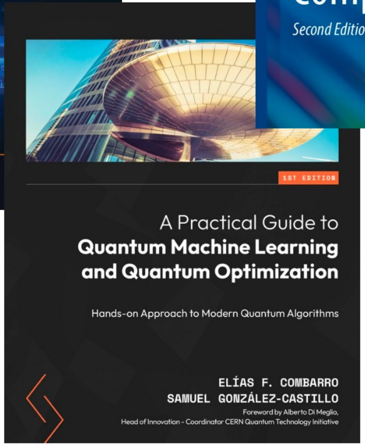
CPUs, GPUs TPUs

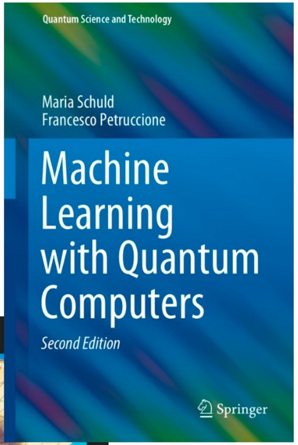
- Inne sieci neuronowe: CNN, AE, GAN, LSTM + QNNs

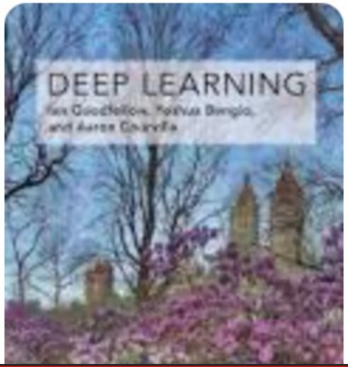
Name	Plot	Function, $g(x)$
Identity		x
Binary step		$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$
Logistic, sigmoid, or soft step		$\sigma(x) \doteq \frac{1}{1+e^{-x}}$
Hyperbolic tangent (tanh)		$\tanh(x) \doteq \frac{e^x - e^{-x}}{e^x + e^{-x}}$
Soboleva modified hyperbolic tangent (smht)		$\operatorname{smht}(x) \doteq \frac{e^{ax} - e^{-bx}}{e^{cx} + e^{-dx}}$
Rectified linear unit (ReLU)[13]		$(x)^+ \doteq \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases} = \max(0, x) = x\mathbf{1}_{x>0}$
Gaussian Error Linear Unit (GELU)[5]		$\frac{1}{2}x\left(1 + \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right)\right) = x\Phi(x)$ where erf is the gaussian error function.
Softplus[14]		$\ln(1+e^x)$
Exponential linear unit (ELU)[15]		$\begin{cases} \alpha\left(e^x - 1\right) & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$ with parameter α
Scaled exponential linear unit (SELU)[16]		$\lambda\begin{cases} \alpha(e^x - 1) & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$ with parameters $\lambda = 1.0507$ and $\alpha = 1.67326$
Leaky rectified linear unit (Leaky ReLU)[17]		$\begin{cases} 0.01x & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$
Parametric rectified linear unit (PReLU)[18]		$\begin{cases} \alpha x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$ with parameter α

Recommended reading on QNN + Deep Learning

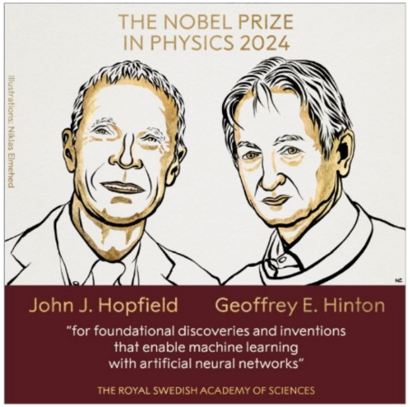


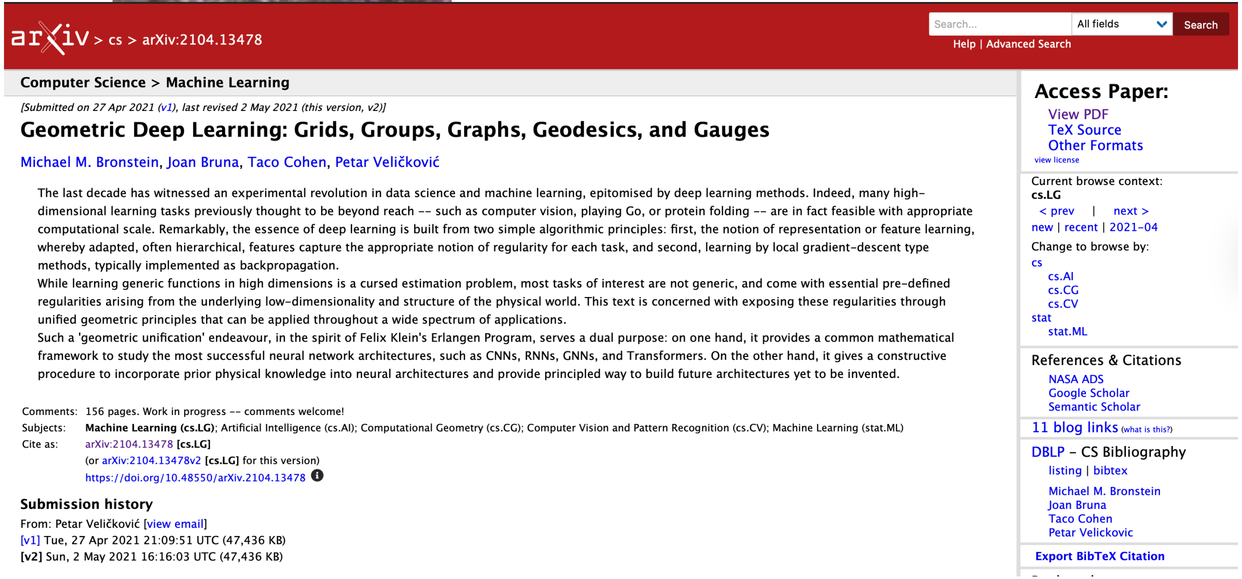






Neural Networks Fathers





arXiv > cs > arXiv:2104.13478

Search... All fields Search
Help | Advanced Search

Computer Science > Machine Learning

[Submitted on 27 Apr 2021 (v1), last revised 2 May 2021 (this version, v2)]

Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges

Michael M. Bronstein, Joan Bruna, Taco Cohen, Petar Veličković

The last decade has witnessed an experimental revolution in data science and machine learning, epitomised by deep learning methods. Indeed, many high-dimensional learning tasks previously thought to be beyond reach -- such as computer vision, playing Go, or protein folding -- are in fact feasible with appropriate computational scale. Remarkably, the essence of deep learning is built from two simple algorithmic principles: first, the notion of representation or feature learning, whereby adapted, often hierarchical, features capture the appropriate notion of regularity for each task, and second, learning by local gradient-descent type methods, typically implemented as backpropagation.
While learning generic functions in high dimensions is a cursed estimation problem, most tasks of interest are not generic, and come with essential pre-defined regularities arising from the underlying low-dimensionality and structure of the physical world. This text is concerned with exposing these regularities through unified geometric principles that can be applied throughout a wide spectrum of applications.
Such a 'geometric unification' endeavour, in the spirit of Felix Klein's Erlangen Program, serves a dual purpose: on one hand, it provides a common mathematical framework to study the most successful neural network architectures, such as CNNs, RNNs, GNNs, and Transformers. On the other hand, it gives a constructive procedure to incorporate prior physical knowledge into neural architectures and provide principled way to build future architectures yet to be invented.

Comments: 156 pages. Work in progress -- comments welcome!
Subjects: **Machine Learning (cs.LG)**; Artificial Intelligence (cs.AI); Computational Geometry (cs.CG); Computer Vision and Pattern Recognition (cs.CV); Machine Learning (stat.ML)
Cite as: arXiv:2104.13478 **[cs.LG]**
(or arXiv:2104.13478v2 **[cs.LG]** for this version)
https://doi.org/10.48550/arXiv.2104.13478

Submission history
From: Petar Veličković [view email]
[v1] Tue, 27 Apr 2021 21:09:51 UTC (47,436 KB)
[v2] Sun, 2 May 2021 16:16:03 UTC (47,436 KB)

Access Paper:
View PDF
TeX Source
Other Formats
view license

Current browse context:
cs.LG
< prev | next >
new | recent | 2021-04
Change to browse by:
cs
cs.AI
cs.CG
cs.CV
stat
stat.ML

References & Citations
NASA ADS
Google Scholar
Semantic Scholar

11 blog links (what is this?)

DBLP - CS Bibliography
listing | bibtex
Michael M. Bronstein
Joan Bruna
Taco Cohen
Petar Velickovic

Export BibTeX Citation

What is
Quantum Technology

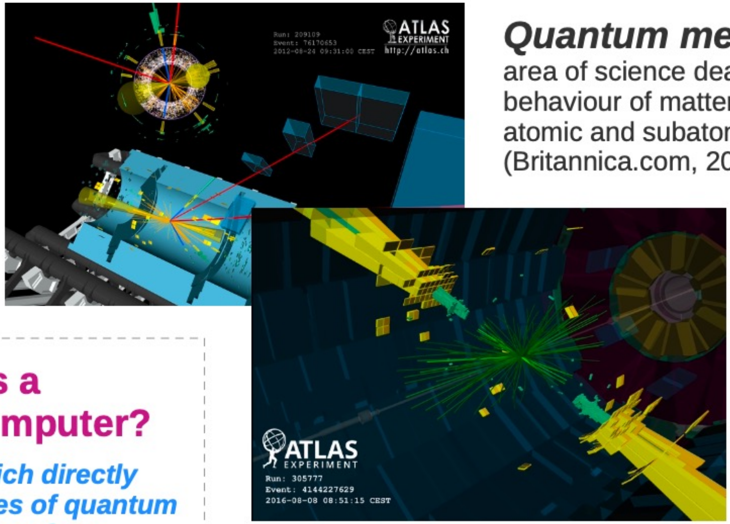
Quantum computing *and*
Quantum information science
study of the information processing tasks that can be accomplished using quantum mechanical systems (Nielsen and Chuang, 2010)

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

$$X \quad \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad Y \quad \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad Z \quad \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

What is a Quantum Computer?

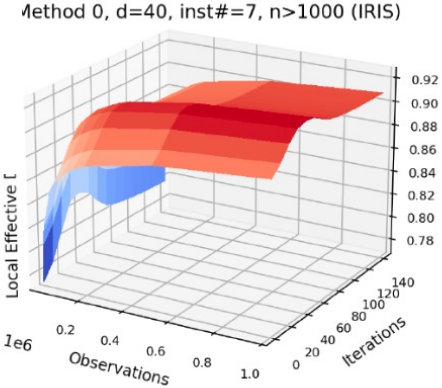
It is a device which directly applies the principles of quantum mechanics to perform computational tasks



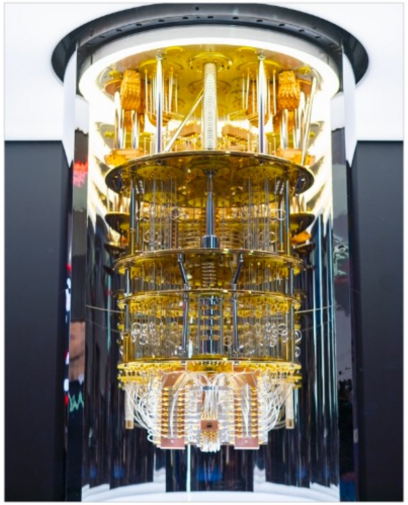
Quantum mechanics
area of science dealing with the behaviour of matter and light on the atomic and subatomic scale (Britannica.com, 2020)

CERN
Atlas particle detector
Large Hadron Collider

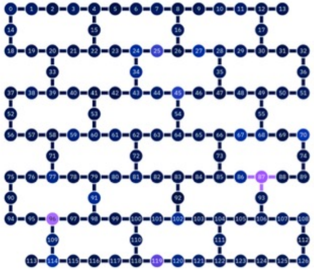
There are other quantum technologies e.g. quantum sensing



Quantum finance
Quantum chemistry
Quantum optimisation
Quantum machine learning
problem-solving with quantum technology

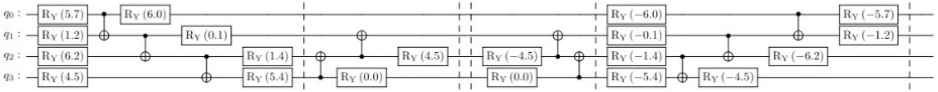


IBM superconducting quantum machine (127 qubits on cloud)



Quantum engineering
building quantum devices

Cybulski & Nguyen Barren Plateaus in QNNs and VQAs



Recently in the news ...

Quantum Machines

(2024)

Univ of Sci and Tech of China with Shanghai Inst of Microsystem and Info Tech (*Jiuzhang 3* - Photonic)

Again achieved quantum supremacy

Gaussian Boson Sampling

SpinQ
(N. Magnetic Resonance)

IQM / VTT
(Superconducting)

IBM (Superconducting)

D-Wave
(Quantum Annealing)

PASQAL
(Neutral Atoms)

Quantum Brilliance
(Diamond)

They all use:

Qubits
the fundamental models of quantum information and its processing

Quantum circuits
models of computation, involving qubits and operations on them

Microsoft Azure Quantum

Providers
- Quantinuum
- IonQ
- QCI
- Rigetti
- Pasqal
- 1QBit
- Microsoft QIO
- Toshiba ...

AWS Braket

(Platforms)

Providers
- IonQ
- Rigetti
- OQC
- Xanadu
- D-Wave
- QuEra ...

Google
(Superconducting)

Xanadu (Photonic)

Qubits

in scientific terms

In practice qubits involve *elementary particles*, such as *ions*, *photons*, single *atoms*, *electrons*, even *defects in diamonds*; and, their behaviour is governed by Physics (Nature / Universe)

A qubit *represents a state* of such a particle, e.g. an electron spin, which can be up or down, (written formally as $|\uparrow\rangle$ and $|\downarrow\rangle$ or $|0\rangle$ and $|1\rangle$), which are called the *basis states*

It is possible to change the state of a qubit with certain predetermined *operations*, such as rotation or reversing the position of the qubit state

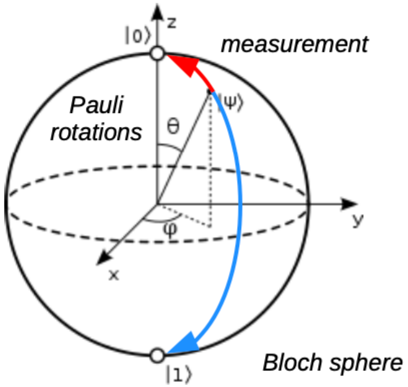
Afterwards, the qubit is in a state of *superposition*, or a combination, of its basis states up and down

The superposition state is the actual state of elementary particles, not its math description

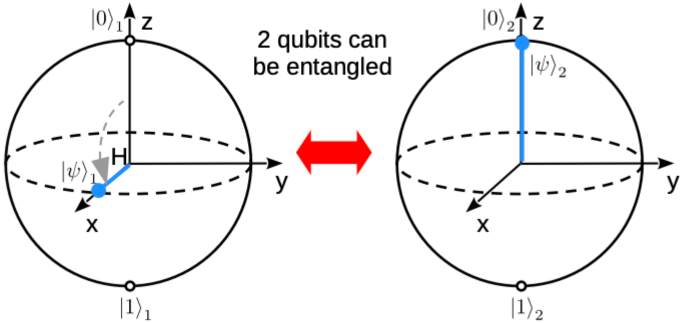
It is impossible to determine the qubit's state without its *measurement*

Qubit measurement returns only the basis state that is "likely" to be closest to its superposition state, which also destroys the qubit state

The outcome of measurement is precise but probabilistic



Qubits can be *entangled*, then they start behaving as a unit with a common complex state, until they are measured or until some external factor (*noise*) destroys their entanglement

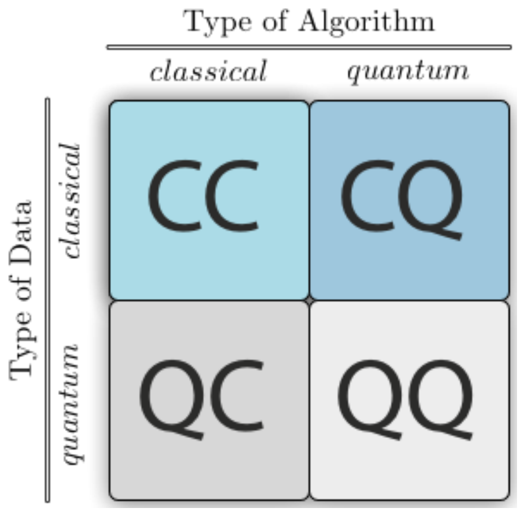


What makes quantum computers special?

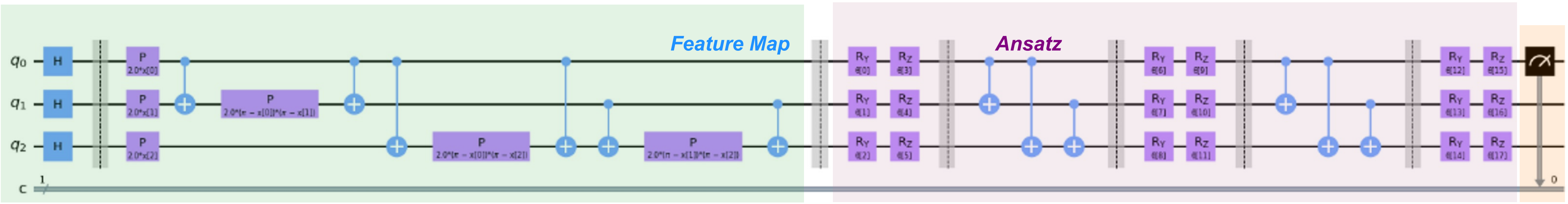
Qubit *superposition* (parallel choices) and *entanglement* (exponential combination of choices and their filtering), as well as *measurement* (collapse of choices and randomness), is what gives quantum computers their immense computational power allowing some problems to be solved in minutes rather than 1000s of years!

Quantum Neural Networks

- QNN są realizowane jako parametryczne obwody kwantowe składające się z:
- *The feature map* warstwa kodująca klasyczne dane
- *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
- *Pomiar* stanu kubitów. Interpretacja otrzymanych wyników.



Kodowanie danych ***“Model”*** ***Pomiar***

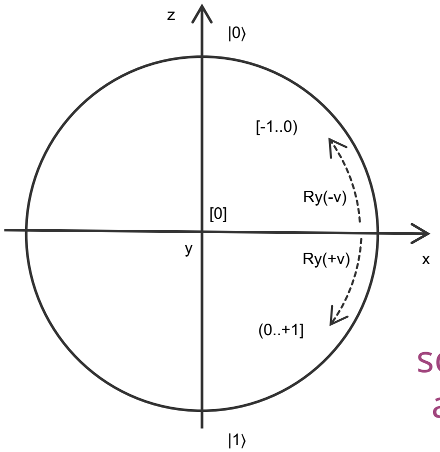


Input encoding / embedding
for QNN processing

Generally, QAE input and output are an unrestricted collection of real values (floats), guiding 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.

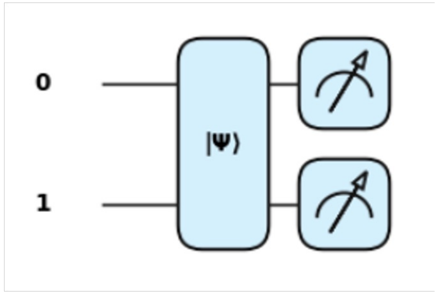


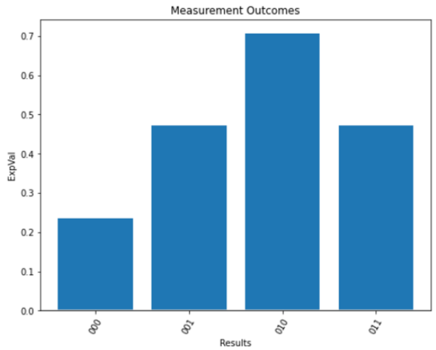
Angle encoding suits QAE design, with input values represented as Qubit state rotations (float).

In our experiments, we used angle encoding relative to |+⟩ state, with values ∊ [-1, 1] scaled to a range [0, π], 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.

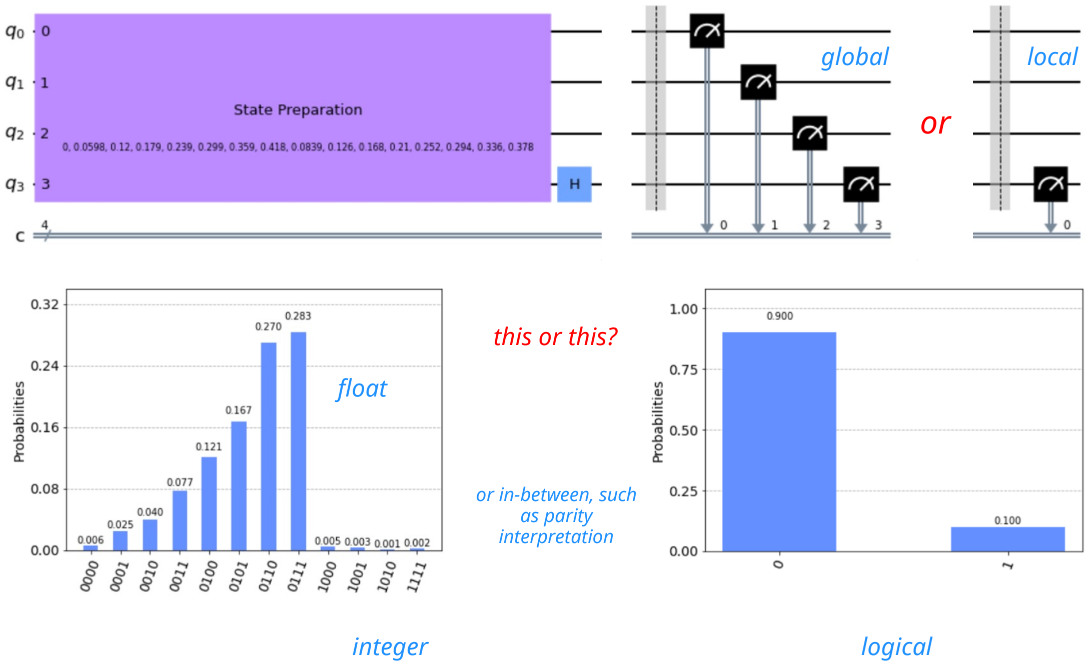
It embeds input as a circuit state normally measured on output, i.e. , each data point is encoded as the expectation value of a multi-qubit measurement (int/*float*).





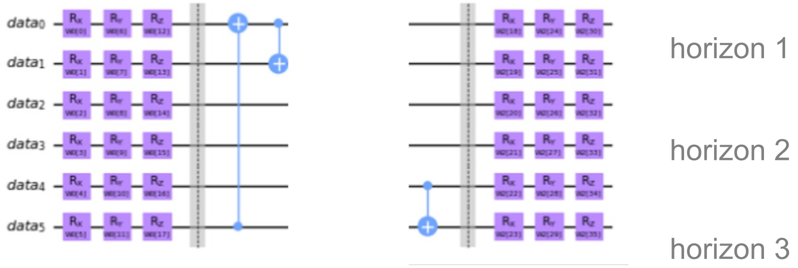
Example: data encoded as ψ was normalised vector [1/8, 2/8, 3/8, 2/8]. The measurement reflects the input data proportions.

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).

Variational Quantum Algorithm

VQA is an ***iterative process***

VQA uses ***cost/loss function*** and ***optimiser***

VQA has ***difficulties***:

- The problem at hand
- Large circuits with many parameters
- Complex measurement strategy
- Unsupervised learning
- Emergence of barren plateaus

A typical VQA process

The ansatz parameters are initialised to some values, e.g. zero or random

The feature map parameters are bound to the new input data

The parameter values are used to create a new circuit

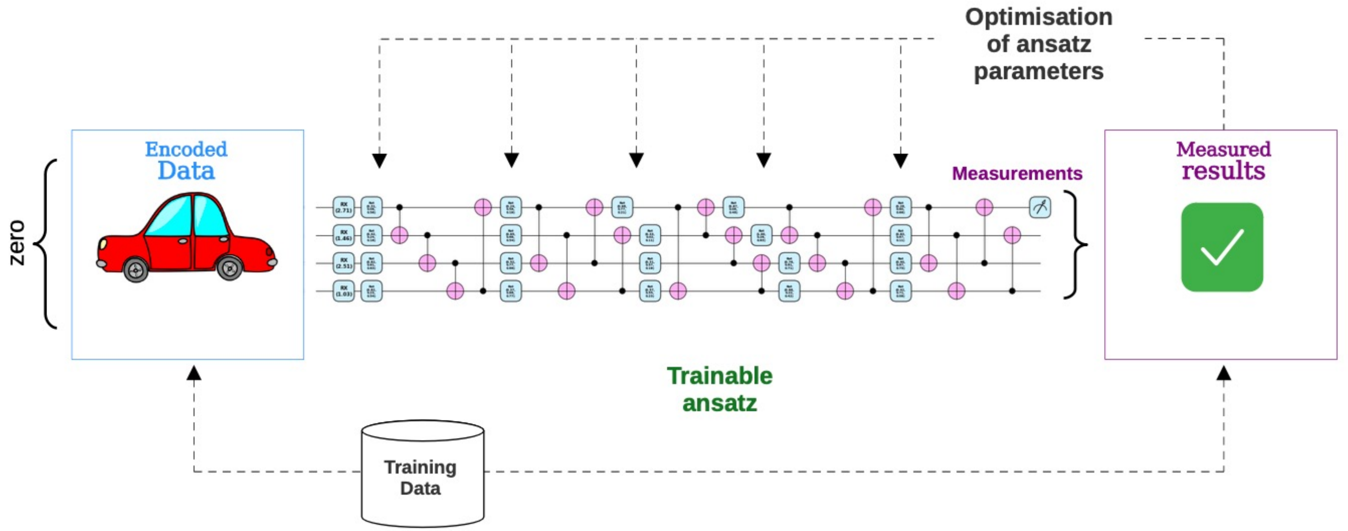
The circuit is executed

The circuit quantum state is then measured

Cost function is applied to measurement results and expected values

The cost of difference is calculated

Based on the difference and previous parameters the new parameters values are proposed

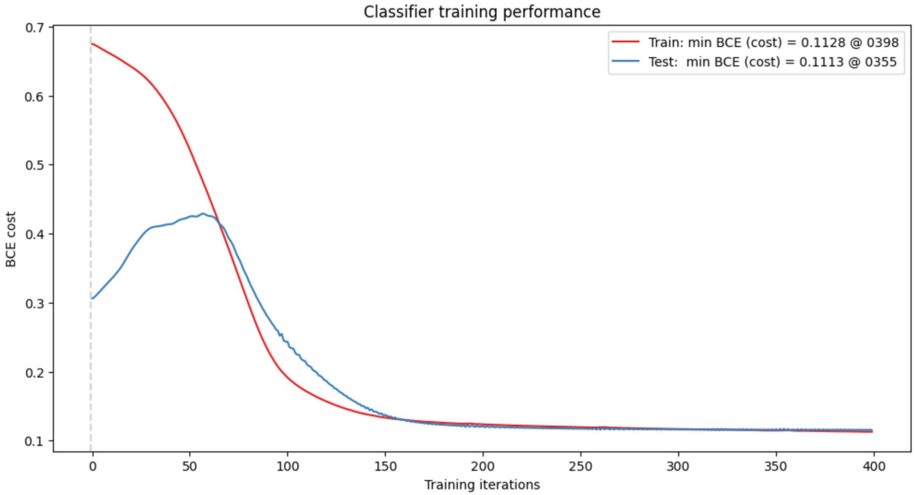


Demo

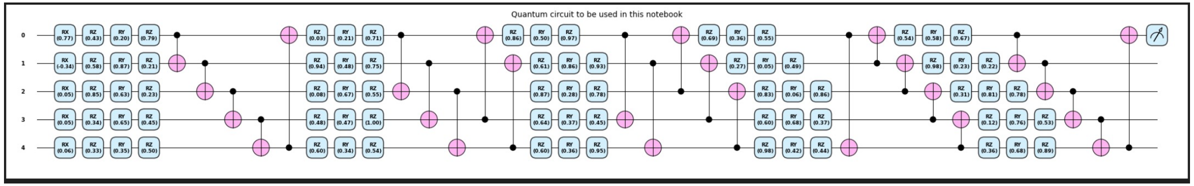
Klasyczna sieć neuronowa

```
CM weights: 3219, Epochs: 400

Classic_NN(
  (model): Sequential(
    (0): Linear(in_features=5, out_features=64, bias=True)
    (1): ReLU()
    (2): Linear(in_features=64, out_features=32, bias=True)
    (3): ReLU()
    (4): Linear(in_features=32, out_features=18, bias=True)
    (5): ReLU()
    (6): Linear(in_features=18, out_features=8, bias=True)
    (7): ReLU()
    (8): Linear(in_features=8, out_features=1, bias=True)
    (9): Sigmoid()
  )
)
```

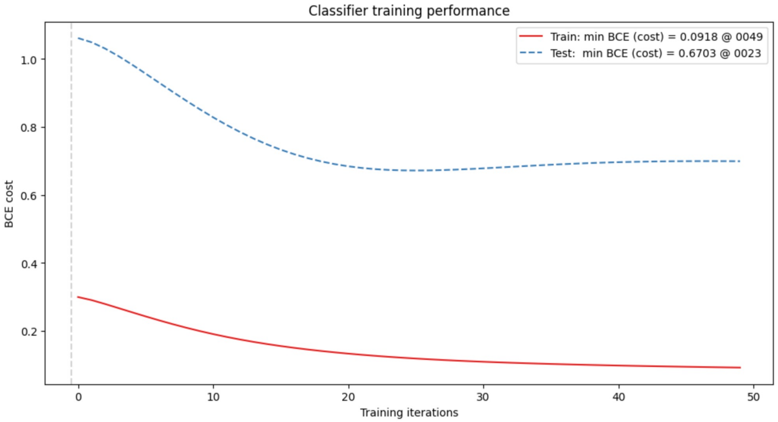


Kwantowa sieć neuronowa




```
QM weights: 75, Epochs: 50

Quantum_Auto(
  (model_pt): Sequential(
    (0): <Quantum Torch Layer: func=_qmodel>
  )
)
```



Quantum Autoencoder (for Time Series)

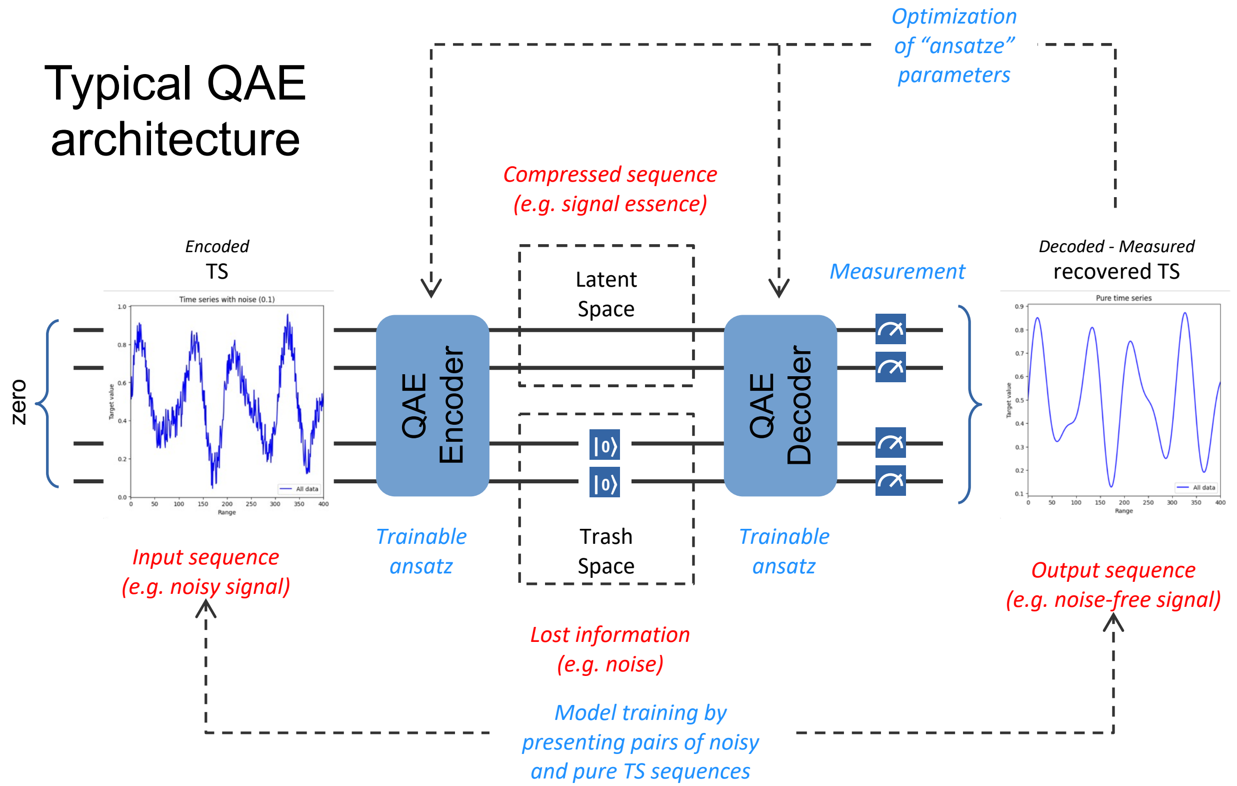
There are a few applications of QML methods to time-series analysis, TS applications of quantum AE (QAE) are even fewer

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

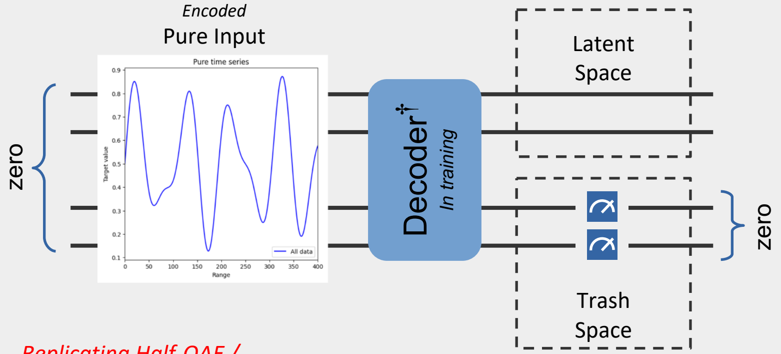
Typical QAE architecture



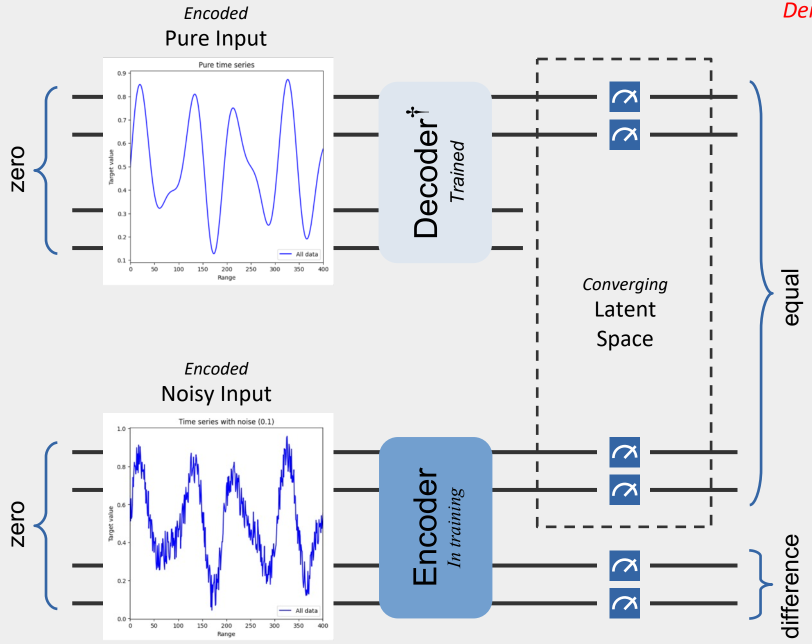
In QAE development, the key concerns include: *overall model architecture, data encoding and decoding, ansatz design and its parameters optimisation strategy*

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

Alternative Architectures



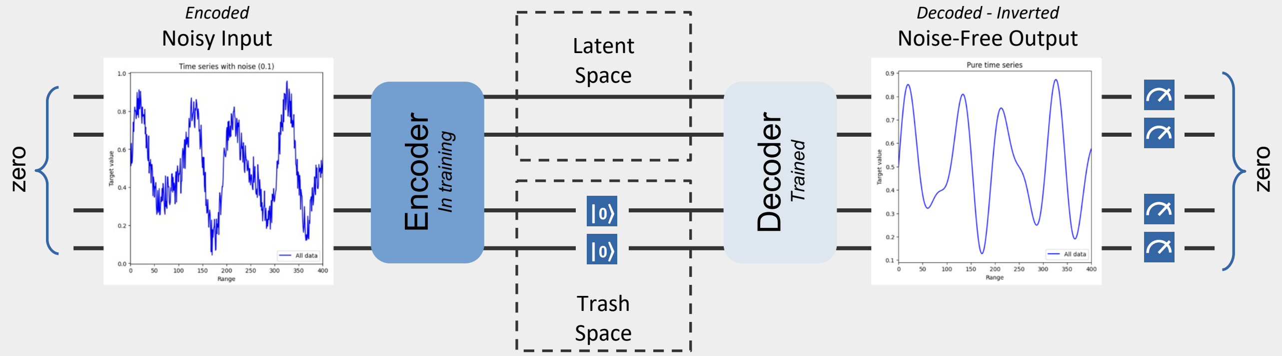
Replicating Half-QAE /
single stage (pure data only)



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

Thank you!