# Quantum Anomaly Detection

## Using generative adversarial networks and quantum technology to identify outliers

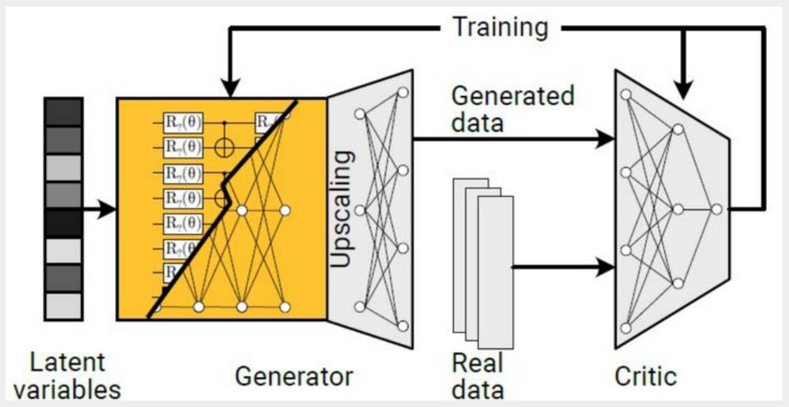
Generative Adversarial Networks (GANs) have several promising applications in anomaly detection. In our work, we replaced the generator part of the GAN with a quantum neural network. Two use cases are presented: credit card fraud detection and fake news detection.

Daniel Herr, Benjamin Obert, Matthias Rosenkranz
d-fine GmbH
An der Hauptwache 7 D-60313 Frankfurt/Main

## Context and Motivation

- Anomaly detection involves the inspection of data points to identify patterns or data points that significantly deviate from the norm or expected behaviour within a dataset. Anomalies, in this context, are data points that do not conform to the typical patterns or distribution of the data and may indicate unusual, rare, or potentially suspicious events or errors.
- Generative Adversarial Networks (GANs) are a class of machine learning models consisting of two neural networks, the generator and the discriminator
  (1) Generator: a neural network that takes random noise or a seed as input and generates data, typically in the form of images (but it can be applied to other data types as well). Its goal is to produce data that is indistinguishable from real data.
  (2) Discriminator: a neural network that acts as a binary classifier. It takes both real data (e.g. real images) and fake data (generated by the generator) as input. Its goal is to correctly classify real data as real and fake data as fake.
- Generative Adversarial Networks are known to represent suitable methods for Anomaly Detection (AnoGAN)
- Our approach: **replace the classical generator with a quantum neural network** (motivated by Google supremacy experiment); NISQ devices are good for sampling

## AnoGAN

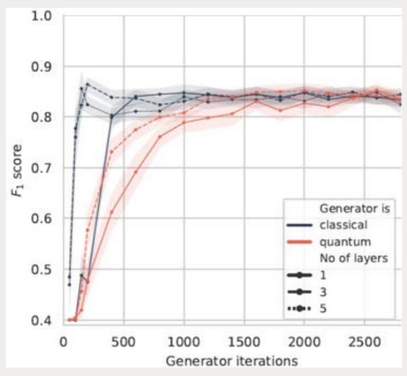



Steps:
1. Train GAN on regular data
2. Find latent variables producing samples closest to unseen data
3. Anomaly score = weighted sum of residual score (difference between sample and generated sample) and critic score (difference between critic outputs of the sample and generated sample)
4. Classify based on anomaly score. Can be extended to feature-level visualisation.

## Use Case: Credit Card Fraud Detection

Kaggle dataset:
- 30 features + label
- Size: ~ 284.000 transactions (over the course of 2 days)
- Highly imbalanced: contamination roughly 0.2%
- Most features are anonymized through principle component analysis (PCA)

- Even with a few qubits, **similarly good results can be achieved** as with classical methods. However, the quantum approach must be trained for longer to achieve this quality of results.
- Running this use case on quantum hardware (IBMQ and Rigetti with Qiskit and Pennylane) is prohibitive due to the large number of circuit evaluations. Batching or Quiskit Runtime is essential




$$Num_{circuits} = Iterations \cdot Batch \cdot DiscriminatorSteps \cdot (\underbrace{2 \cdot num_{qubits} \cdot depth}_{\text{parameter shift rule}} + 1)$$

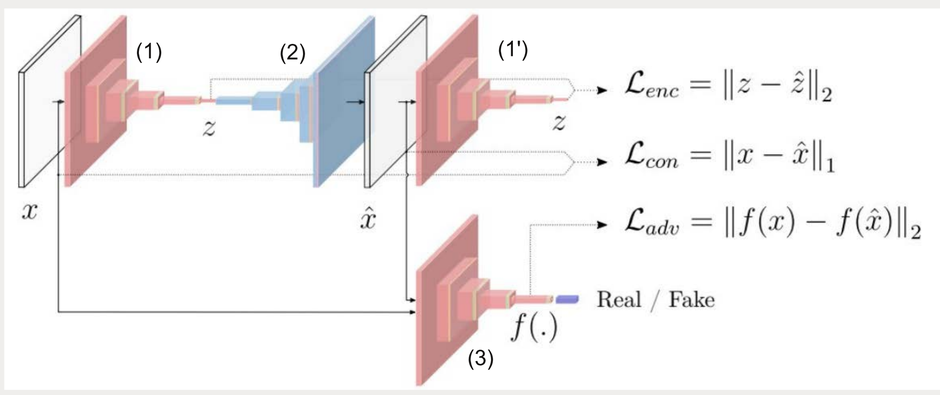
- Training steps: 4000
- Batch size: 32
- Discriminator steps: 4
- Parameter shift rule: 72

→ **Total circuits: ~37 M**
→ **Estimated time: ~60 years**

## GANomaly [01]

- Extension to previous anomaly detection method: auto encoder structure
→ considerable time savings for the inference

- To enhance the interaction between the paragraph embedding dimensions, multiple dimensions can serve as input for the same qubit
- 600 training steps each training-prediction run (both classical and quantum)
- Issues with high-dimensional data (latent dimension: 50, NLP vector dimensionality: 10). Quantum realization of the generator used 15 distinct circuits each consisting of 10 multi-input qubits.




- Current setup does not assume any structure among the data
  (1) Keras Model with 4 dense layers with LeakyReLu, reducing size from input_dim to latent_dim. So far the encoder (1') has been of the same structure as (1)
  (2) Keras Model with 4 dense layers with LeakyReLu, increasing size from latent_dim to input_dim
  (3) Keras Model with 2 dense layers with LeakyReLu and a single dense layer of size 1 without activation

## Use Case: Fake News Detection

- News are classified as fake if their anomaly score exceeds a trained threshold
  - Threshold was optimized retrospectively with respect to labelled data of the training set
- Runs (training and prediction) were done 35 times independently for Classical- and Quantum-GAN respectively

- Anomaly score is built on latent features: $A(x) = z(x) - \hat{z}(x)$
- Explainability still possible with a comparison of $x, \hat{x}$

## Outlook

Investigate ways to speed up the optimization:
- Forward-Forward Algorithm [02]: replace the forward and backward passes of backpropagation by two forward passes
- Gradient-free optimization methods

Potential extensions:
- Images: replace encoder, decoder by convolutional networks
- Time series: recurrent neural networks, LSTMs

**References:** [01] GANomaly, https://arxiv.org/abs/1805.06725 [2] https://arxiv.org/abs/2212.13345

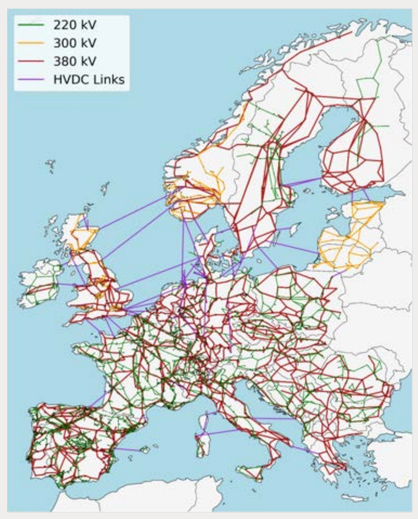
# Cost Optimization of Energy Grids

## Using quantum technology to improve the performance of solvers

The transition of the energy supply structure in Germany and the EU present several challenges for the power grid configuration. Several promising approaches involving quantum technology exist to perform cost optimization.

Oliver Siccha, Daniel Ohl de Mello, Benjamin Obert, Daniel Herr
d-fine GmbH
An der Hauptwache 7 D-60313 Frankfurt/Main

## Context and Motivation




**Situation**
- Energy transition in Germany and the EU
  → radical transformation of energy supply
  → challenges for generators and network operators

**Goals**
- ensure reliability of supply
- develop renewable energy sources
- keep costs under control

**Challenges**
- increased volatility
- efficient network expansion
- changed energy consumption

→ Forecasting model for power generation, demand and transport by optimization of investment and operating costs using open-source frameworks like PyPSA

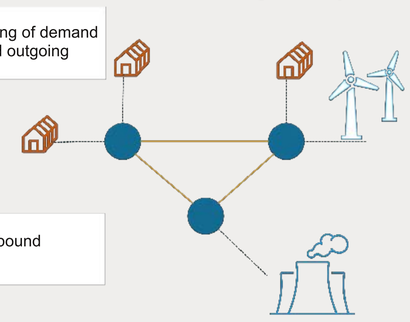
## Unit Commitment Problem

Which power plants to use to cover demand?
- Deployment: power plants are in a state (on/off)
- Rampup/Rampdown: gradual change in power during ramp-up/-down
- Cost of startup/shutdown
- Up-/Downtime: power plants have minimal downtime

Modelling in PyPSA

**Busses**
Kirchhoff's law: exact covering of demand by locally generated, in- and outgoing electricity

**Demand**
Temporal variation to be covered at any point in time



**Cables**
Variable flow with absolute bound

**Generators**
Variable power with generator-dependent upper bounds

$$\min_{g_{n,s,t}} \sum_{n,s} \underbrace{c_{n,s}\bar{g}_{n,s}}_{\substack{\text{power plant}\\ \text{development}\\ \text{(capital cost)}}} + \sum_{n,s,t} \underbrace{o_{n,s,t}g_{n,s,t}}_{\substack{\text{power plant}\\ \text{operation}\\ \text{(operational cost)}}} + \sum_{l} \underbrace{c_l F_l}_{\substack{\text{cable}\\ \text{development}\\ \text{(capital cost)}}} + \sum_{n,s,t} \underbrace{(suc_{n,s,t} + sdc_{n,s,t})}_{\substack{\text{startup/shutdown costs}\\ \text{(operational cost)}}}$$

| | |
|---|---|
| $c$ | capital costs |
| $o$ | operational costs |
| $\bar{g}$ | max power of plant (can be optimized) |
| $g$ | power of plant (can be optimized) |
| $F$ | max transmission power (can be optimized) |
| $suc$ | startup cost |
| $sdc$ | shutdown cost |

## Unit Commitment as a QUBO Problem

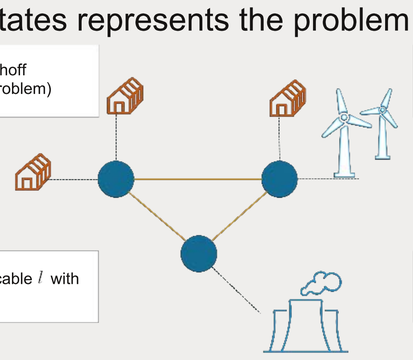
- QUBO: Quadratic Unconstrained Binary Optimization

$$\min f(x_1, ..., x_n) = \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}x_i x_j, \quad x_i \in \{0, 1\}$$

- Each variable describes the state (generators) or electricity flow (cables) at a certain point in time
- Each state has a value w.r.t. each measurand (electricity, cost)
- The sum of all states represents the problem at a certain point in time

**Busses**
One QUBO per bus for Kirchhoff condition (constraint of the problem)

**Demand**
$d_{B,t}$ : demand at bus $B$ at point in time $t$ (problem-dependent constant)



**Cables**
$x_{l,t,1}, ..., x_{l,t,2c}$ : flow through cable $l$ with capacity $c$ at point in time $t$

**Generators**
$x_{g,t}$ : state of generator $g$ at point in time $t$ (optimization variable)

$$QUBO_{total} = \sum_{constraints} weight_{constraint} \cdot QUBO_{constraint} + QUBO_{costfunction}$$

- Quadratic distance - Kirchhoff condition

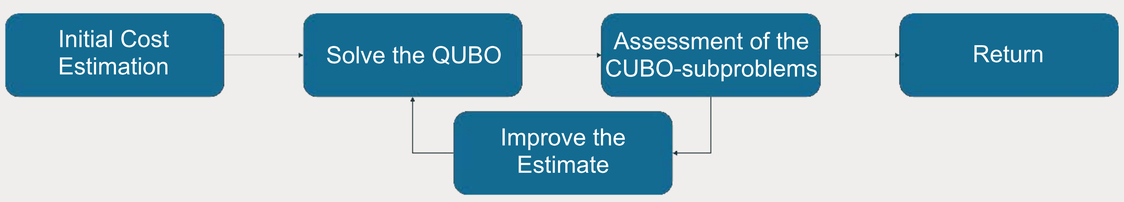
$$QUBO_{constraint} = \left(-d_B + \sum_{i=1}^{n} w_i x_i\right)^2$$

| | |
|---|---|
| $x_i$ | polynomial variables |
| $w_i$ | corresponding power values |
| $d_B$ | demand at bus $B$ |

$$x_i \in \{0,1\} \implies -d_B w_i x_i = -d_B w_i x_i^2$$

- Quadratic distance - cost function

$$QUBO_{costfunction} = \left(-estimate + \sum_i cost_i \cdot x_i\right)^2$$

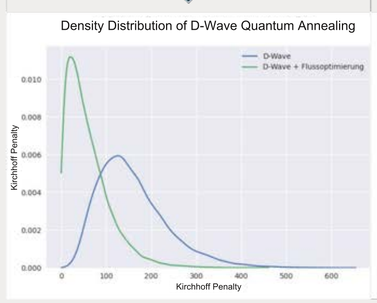



→ formulation of the optimization goal as quadratic distance to an estimate allows for iterative improvement of the QUBO-formulation

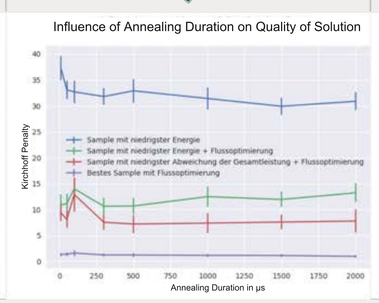
## Solver: Quantum Annealing

- Solving using quantum hardware
- Annealer: D-Wave Advantage 4.1
- Limitation by NISQ-hardware
- Noise
  - Error correction by flow optimization
  - more shots for feasible solutions
- Embedding into working graph
  - Overhead grows with problem size
  - more constraints

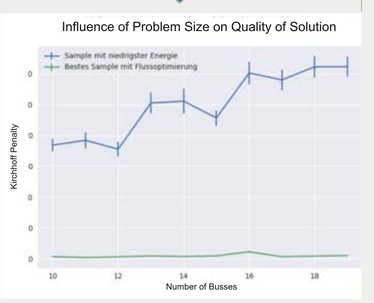

**Distribution as anticipated**




**Annealing duration quickly becomes irrelevant**

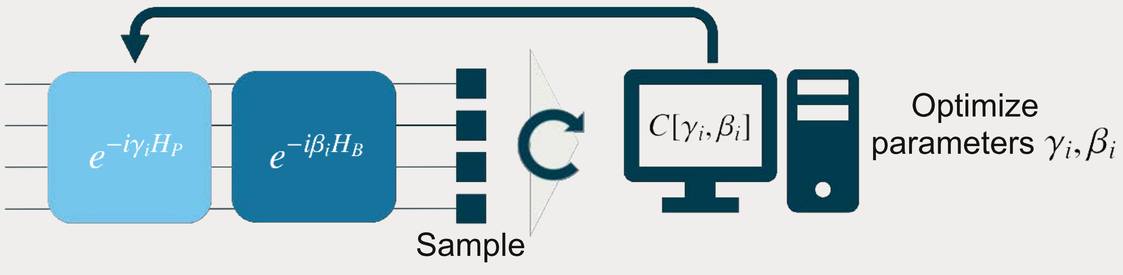



**Error grows with problem size**

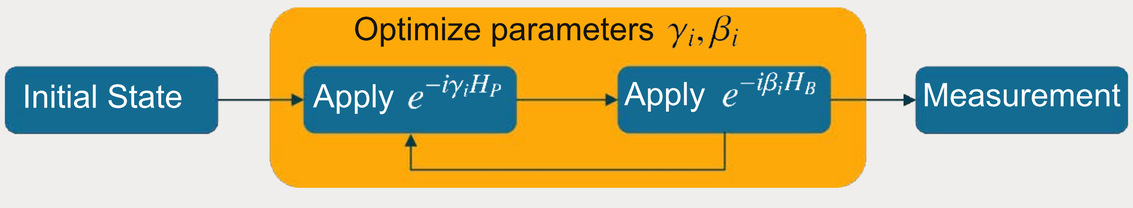



## Solver: Quantum Approximate Optimization Algorithm (QAOA)

- Classical optimization of a quantum circuit
- Limiting the problem size



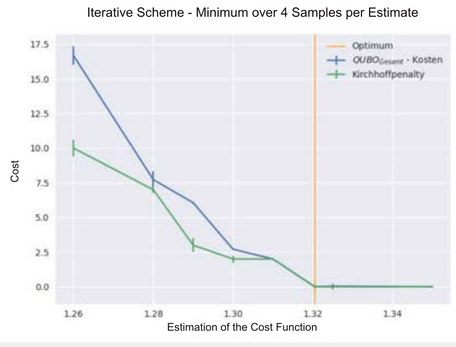

- Initial choice of parameters is important



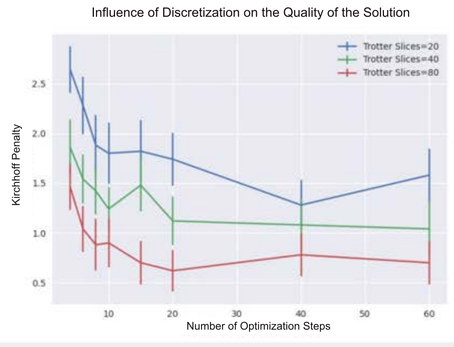

## Solver: Simulated Quantum Annealing (SQA)

- Assessment of QUBO independent of hardware limitations
- Forecast for quantum annealing for hardware progress
- Linear programs can be solved exactly, therefore SQA must deliver better runtime for feasible solutions

**Quality of the Solution**
- Solutions are **feasible**
- Iterative SQA ansatz yields good results
- Requires good solution of subproblems




**Runtime**
- Dependent on solver parameters
- Trotter Slices have biggest influence



# Quantum Deep Hedging

## Exploring the potential of QNNs for mitigating market risks

Hedging is used universally in the financial industry. Deep Hedging involves the application of deep neural networks to hedging. Quantum Computing and in particular Quantum Neural Networks have several promising applications in finance. This contribution explores their performance in terms of speed and accuracy for hedging. Results suggest an advantage of the quantum technology over classical approaches.

Hendrik Heine, Daniel Ohl de Mello, Simon Wittum, Oliver Siccha, Daniel Herr
d-fine GmbH
An der Hauptwache 7 D-60313 Frankfurt/Main

## Context and Motivation

- Consider the following problem:
  - We have sold a derivative (e.g. a portfolio of call options for a stock)
  - Goal: Minimize risk by trading the underlying asset
  → we want to hedge the derivative
- Deep Hedging: classical deep neural networks applied to hedging [01]
- Idea: Utilize the capability of a neural network to act as a universal function approximator
- Goal: Minimize the loss function [02] $\min: L[Z_T + a*H_T - C_T(a)]$

with $a*H_T := \sum_{r=t}^{T-1} a_r(H_T - H_r)$ and $C_T(a) := \sum_{r=t}^{T-1} c_r(a_r)$

$Z_T$: liability amount at $T$, $H_r$: model price of stock at $r$,
$a$: transaction cost at $r$, $c_r$: number of traded units at $r$

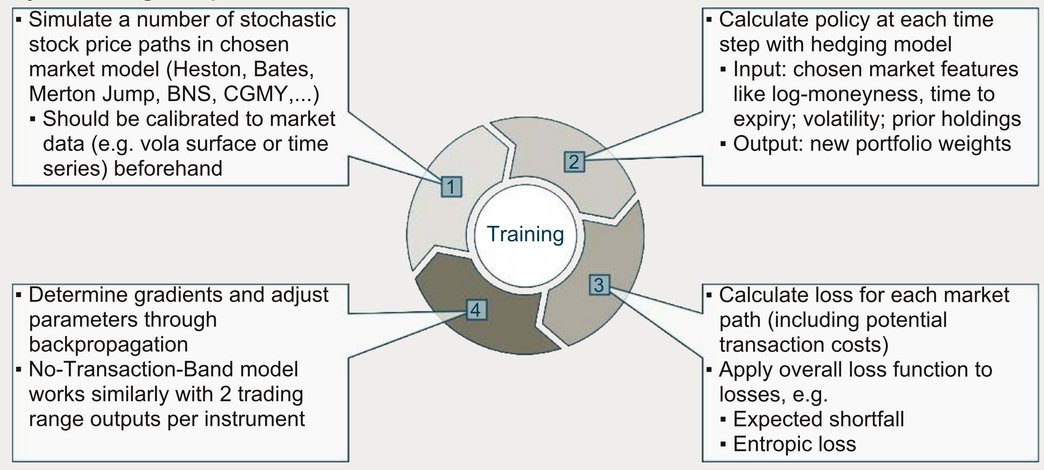
- Loss over different market paths are evaluated with a convex risk measure
  - Our choice: 10% expected shortfall

## Classical Deep Hedging

- Idea: The parametrized loss function can be mapped onto a recurrent neural network $\min_\theta: L[T_T + a(\theta)*H_T - C_T(a(\theta))$

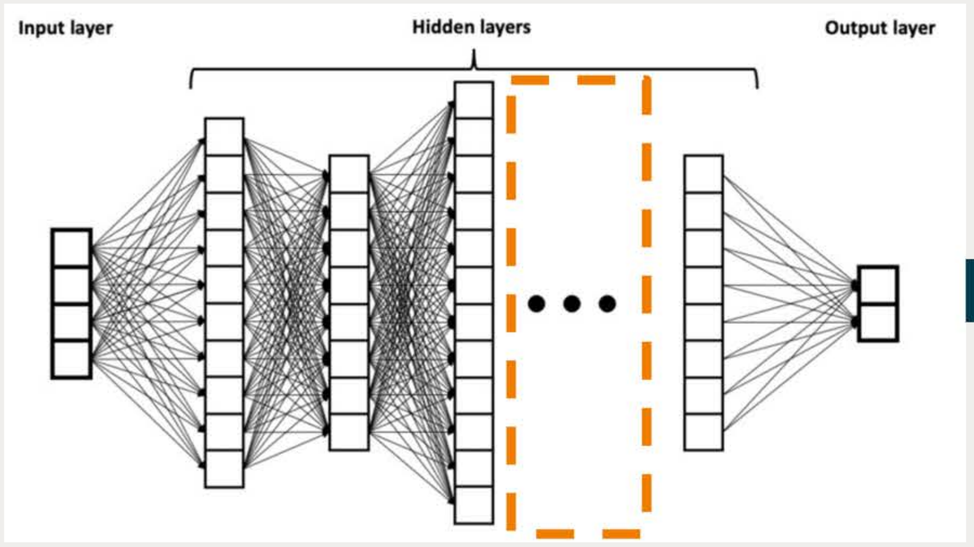
with $\theta = (w_0, ..., w_{K-1}; b_0, ..., b_{K-1})$

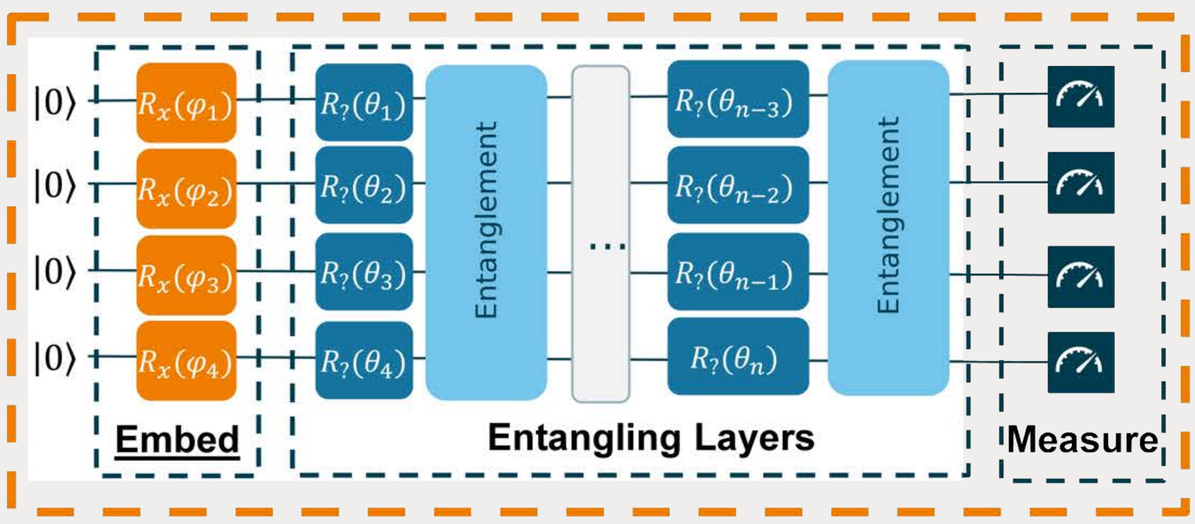
- Key training steps:




## Concepts and Implementation

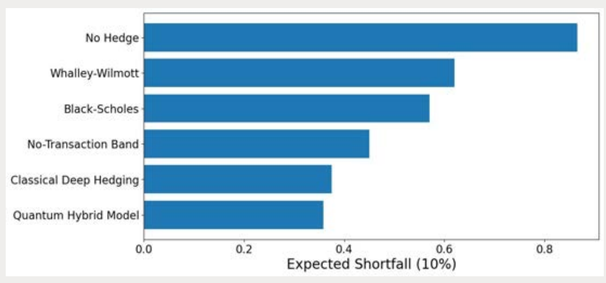
- Idea: Replace (parts of the) classic neural net with a trainable quantum circuit
- Weights and biases get mapped to parameters by unitary operators (e.g. generic rotation gates)
- Loss for a given parameter set evaluated by decoding or measurement of a subset of qubits
- Approach relatively robust against noise → suited for NISQ devices
- But: trade-off between expressive power and trainability should be kept in mind



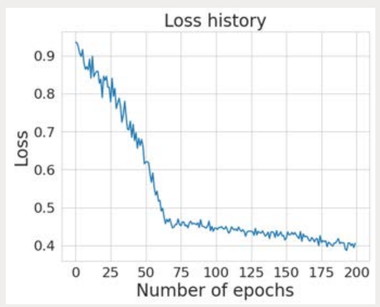




## Results - Classical vs. Quantum

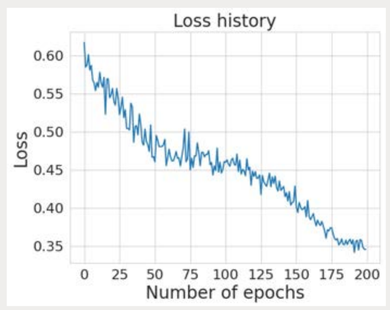
### Performance comparison of hedging approaches




Classical




Quantum




- Adding a quantum layer to a model can help performance
  But: Additional classical layers can accomplish the same with much less training
- Find advantage by putting limits to the models

### Classical vs. Quantum in for few parameters and reduced training epochs

Example price paths


- A distribution featuring large jumps in either direction, causing problems for traditional approaches

Classical







Quantum




- Quantum models have an advantage in this few-parameter environment

## Outlook

- Run the circuits on actual hardware
- Vary the ansatz for the embedding and entangling layers: test other rotation gates
- Test more instruments and derivatives
- Test hedging with combination of derivatives and underlying
- Quantum Monte-Carlo

## Technical Notes

- JAX interface to Pennylane allows for evaluation of circuits without simulations and automatic differentiation of circuits
- Flexible batching and just-in-time compilation of circuits and gradients
- Wrote a Pytorch translation layer which
  - does all the forwards/backwards work in JAX
  - uses helper functions to translate into/from JAX arrays

→ JAX delivers major speedups (factor 1000) compared to directly connecting Pennylane to Pytorch

**References:** [01] arXiv:1802.03042 [02] adapted from https://ssrn.com/abstract=4151041

# eXplainable AI for Quantum Machine Learning

## Adapting classically well-established xAI methods for the quantum use case

Parametrized quantum circuits enable a novel method for machine learning but also present a challenge to existing eXplainable AI (xAI) methods. For example, the state space scales exponentially with the number of qubits, and quantum measurements introduce probabilistic errors that impact the convergence of xAI methods. In this work, we discuss the performance of established xAI methods when applied on quantum hardware, and we present ways to speed up their computation.

Patrick Steinmüller, Tobias Schulz, Ferdinand Graf, Daniel Herr
d-fine GmbH
An der Hauptwache 7 D-60313 Frankfurt/Main

## Context and Motivation

- Most machine learning models are too complex for a proper interpretation
  → eXplainable AI (xAI) methods have been devised to study these models
  → xAI helps in understanding how predictions are obtained
  → Importance of xAI is highlighted by, e.g., governmental initiatives, which require some level of transparency for models in high-risk areas

- Quantum circuits challenge currently available xAI methods:
  1. Quantum hardware inevitably introduces noise to the model.
  2. The state space of a quantum circuit scales exponentially.

- Here, we apply xAI methods to Parametrized Quantum Circuits (PQCs):

  A PQC with $N$ qubits and $n+1$ features consists of $n+1$ single-qubit rotation gates $R^j_\bullet(\varphi_i)$, as well as two-qubit entanglement gates. Here, $j = 1,\dots,N$ denotes on which qubit the gate acts on, $\bullet = x, y, z$ is a placeholder for the specific axis of rotation, and $i = 1,\dots,n+1$ indexes the parameters.

- One can show [1] that the output function $f(\varphi)$ of a PQC is given as a truncated Fourier series, i.e., a continuously differentiable function
- In the following, $f$ denotes a model, $x$ the input value for which an explanation is sought, and $b$ a base value.

## Two Model-Agnostic xAI Methods

- In our work, we focus on the following two currently available, model-agnostic (black-box) xAI methods:
  - **Integrated Gradients (IG)** [2]:

$$\mathrm{IG}(e) = \int_0^1 \frac{\partial f}{\partial x_e}(\gamma(s))\frac{\mathrm{d}\gamma_e}{\mathrm{d}s}(s)\mathrm{d}s \approx \frac{x_e - b_e}{2N\delta_e}\sum_{i=1}^{N-1} f(\gamma_{N;i}+\delta_e) - f(\gamma_{N;i}-\delta_e)$$

  with the axis of interest $e$, the $N$ mesh $\gamma_{N;i} = xi/N + b(1-i/N)$ where $i = 1,\dots,N$, and the shift $\delta_e$.

  - **Baseline SHAP (BS)** [3]:

$$\mathrm{Sh}(e) = \frac{1}{n+1}\sum_{S\subseteq F\setminus\{e\}} \frac{|S|!(n-|S|)!}{n!}[f(g_{S\cup\{e\}}) - f(g_S)]$$

  with the axis of interest $e$, the set of features $F = [n+1]$, the subset $S \subseteq F\setminus\{e\}$, and the intermediate vector $g_S$ defined as $(g_S)_i = \begin{cases} x_i, & i \in S \\ b_i, & i \notin S \end{cases}$.

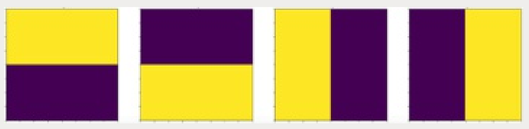
- Both IG and BS are path-dependent feature-perturbation methods and linear in $f$
- Implementation on quantum hardware requires many function evaluations

## Introducing qSHAP

- Making use of the explicit form of the model for PQCs, we develop a (faster) extension of BS for PQCs, which we call **qSHAP**. It consists of three steps:
  1. Evaluate the PQC on random sample points.
  2. Compute the Fourier decomposition of the output function and approximate it as Taylor polynomial.
  3. Use the PolynomialSHAP algorithm [4] to compute the contribution of each term.
- PolynomialSHAP [4] extends the concept of LinearSHAP to multivariate polynomial functions
- qSHAP does not scale exponentially in the number of qubits, but rather in the number of features

## Experiment Setup

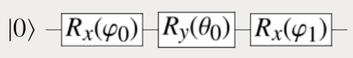
- To compare the different xAI methods when applied to PQCs, we employ the *2 by 2 Bars and Stripes Problem*:

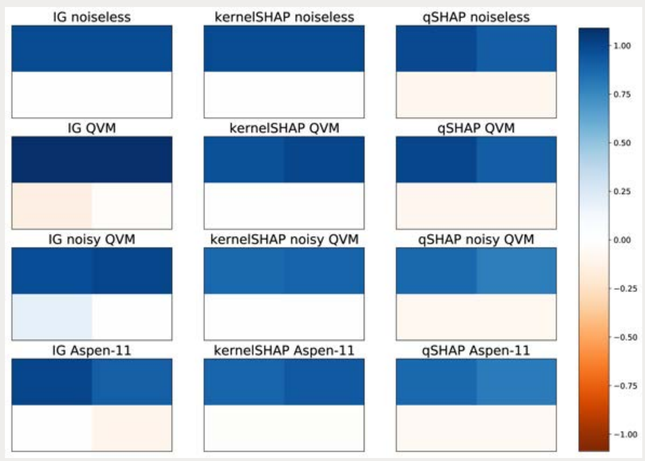


- Aim: Distinguish between the images of bars and images of stripes.
- We classically trained three different classifiers, each using a different PQC
- We compare qSHAP with IG and KernelSHAP (average of BS over several base vectors) across different backends:
  - cl. simulation with no noise (except for shot noise)
  - cl. simulation with a generic noise model
  - cl. simulation with a realistic noise model of the quantum hardware
  - Rigetti's quantum computer (Aspen-11)

## Experiment Results

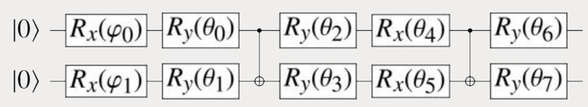
Single-Qubit Classifier (two features; top pixels):

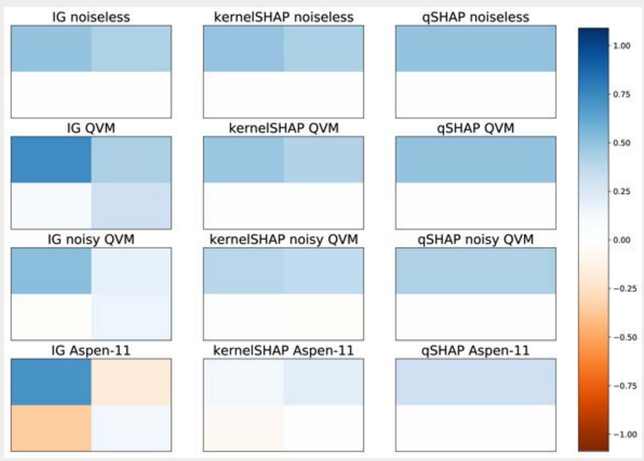



- Top pixels dominate the model prediction
- Noise has biggest impact on IG
- qSHAP is most robust against noise

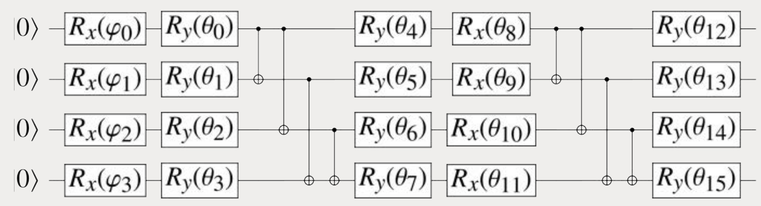
Two-Qubit Classifier (two features; top pixels):

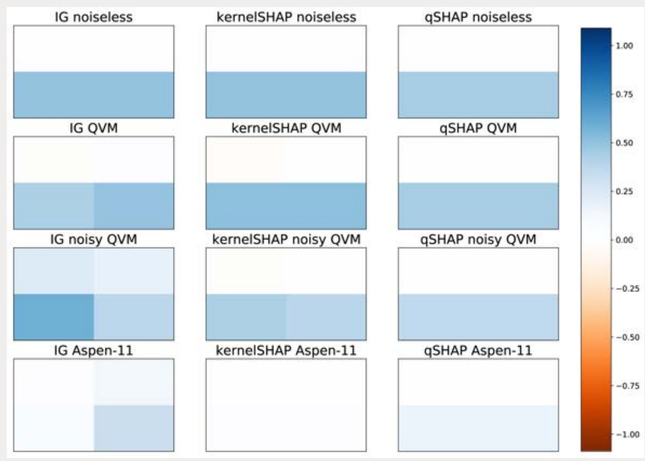



- Different xAI models behave similarly
- Noise has less influence on this classifier, but biggest impact on IG
- Explainability of bottom pixels due to noise

Four-Qubit Classifier (four features; all pixels):




- All pixels could affect the classification
- Noiseless case: Only bottom pixels affect the classification
- Noise seems to have least effect on qSHAP

## Summary, Conclusion and Outlook

- We applied two black-box xAI methods to quantum machine learning methods
  - Drawbacks: Exponential scaling and behaviour under quantum noise
    - First steps towards solution: We develop qSHAP
      → xAI method specifically designed for PQCs
- For qSHAP, the adverse effect of noise is considerably reduced compared to the other xAI methods

Outlook:

- Training the classifier on the actual backend optimizes the model
- We expect new approaches due to the overlap between xAI for PQCs and the general simulation of quantum circuits
- Future work could focus on finding efficient algorithms for approximating PQCs to further improve the qSHAP algorithm

**References:** [1] Phys. Rev. A 103, 032430 (2021) [2] PMLR 70:3319-3328 (2017) [3] arXiv:1705.07874 (2017) [4] arXiv:2211.01441 (2022)