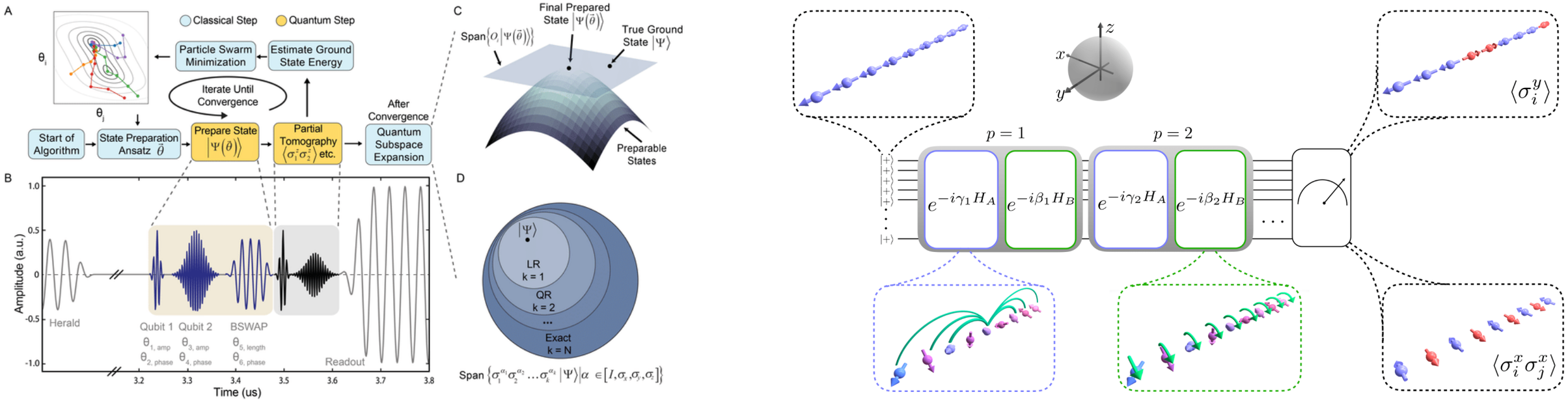


Variational Quantum Algorithms

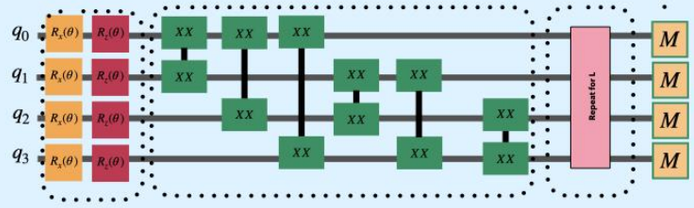
Variational Quantum Algorithm (VQA)

Classical Pre-Processing

$\mathbf{x} \sim P_D$

$\mathbf{x} \longrightarrow \phi(\mathbf{x})$

Quantum Computer



Classical Post-Processing

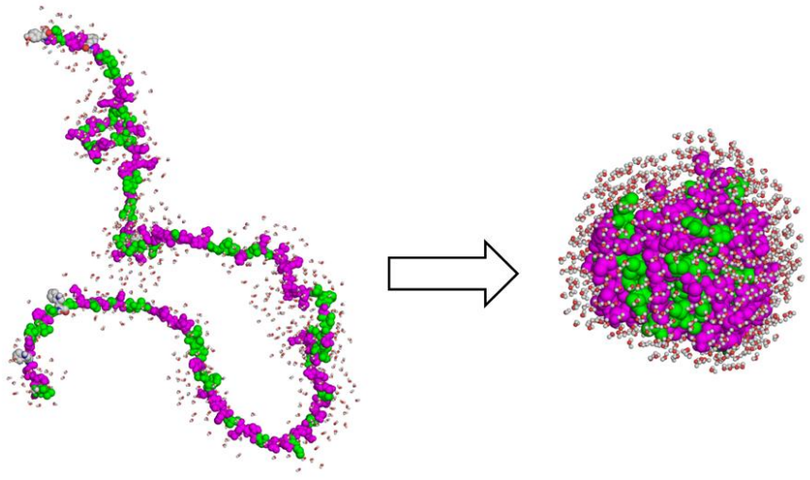
$\{\langle M_k\rangle_{\mathbf{x},\theta}\}_{k=1}^{K}$

Minimize

$L(f(\theta))$ ····· $f(\{\langle M_k\rangle_{\mathbf{x},\theta}\}_{k=1}^{K})$

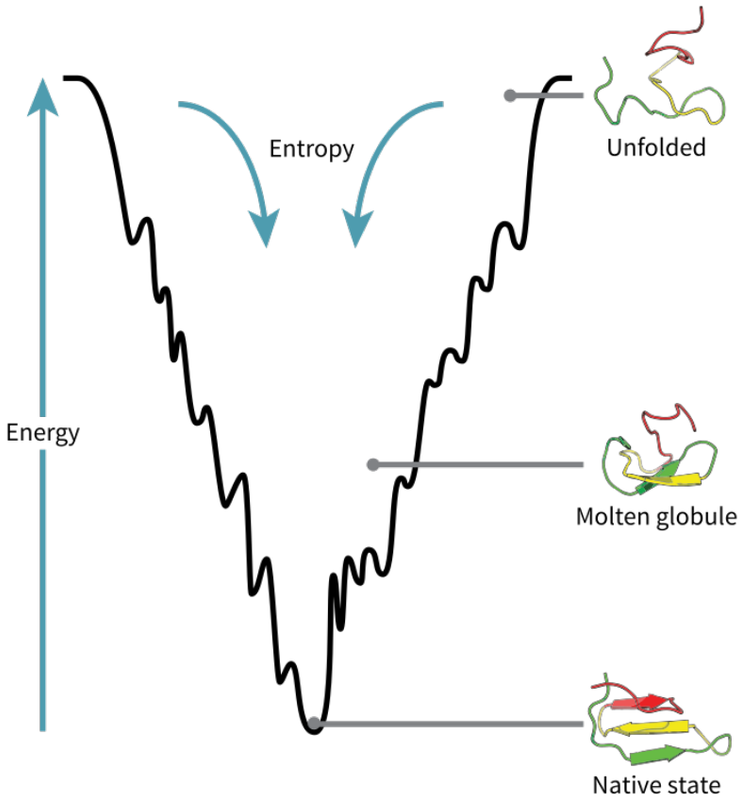
Update $[\theta_0, \theta_1, \theta_2 \ldots \theta_p]$ until optimal.

Protein-folding and Levinthal's Paradox



Unfolded Folded

- Elongated proteins fold to same state within microseconds
- Some proteins have 3^{300} conformations
- Levinthal's Paradox (1969):
  Sequential sampling of states would take longer than lifetime of Universe (even if only nanoseconds per state spent)



- Solution: No sequential sampling, but rapid descend into the potential minimum. In proteins due to protein folding intermediates

→ Optimisation = Life

→ Solution of mathematical problem can be found quickly if encoded in ground state of complex system

Variational Quantum Eigensolver (VQE)

- Initially proposed to find ground state of quantum system
- QM tells us that the ground state $|\psi\rangle$ minimises the expectation value of the Hamiltonian, i.e. the energy $\langle\psi|H|\psi\rangle$
- Note, H is time independent. Idea is to parametrise the state preparation using an ansatz $W(\theta)$ such that $|\Psi(\theta)\rangle = W(\theta)\,|0\rangle$

  $\longrightarrow$ use as cost function $C(\theta) = \langle\psi(\theta)|H|\psi(\theta)\rangle$

  $\longrightarrow$ instead of ground state, we find parameters θ that minimise cost function

- One of the biggest challenges of VQE is to encode the physical system's Hamiltonian. Fortunately, H can often be expressed as sum of local operators

$$H = \sum_{j=1}^{J} h_j H_j \quad \xrightarrow{\text{Pauli operators}} \quad \langle H\rangle = \sum_{i,\alpha} h^i_\alpha \langle\sigma^i_\alpha\rangle + \sum_{i,j\alpha,\beta} h^{ij}_{\alpha,\beta}\langle\sigma^i_\alpha\sigma^j_\beta\rangle + \cdots$$

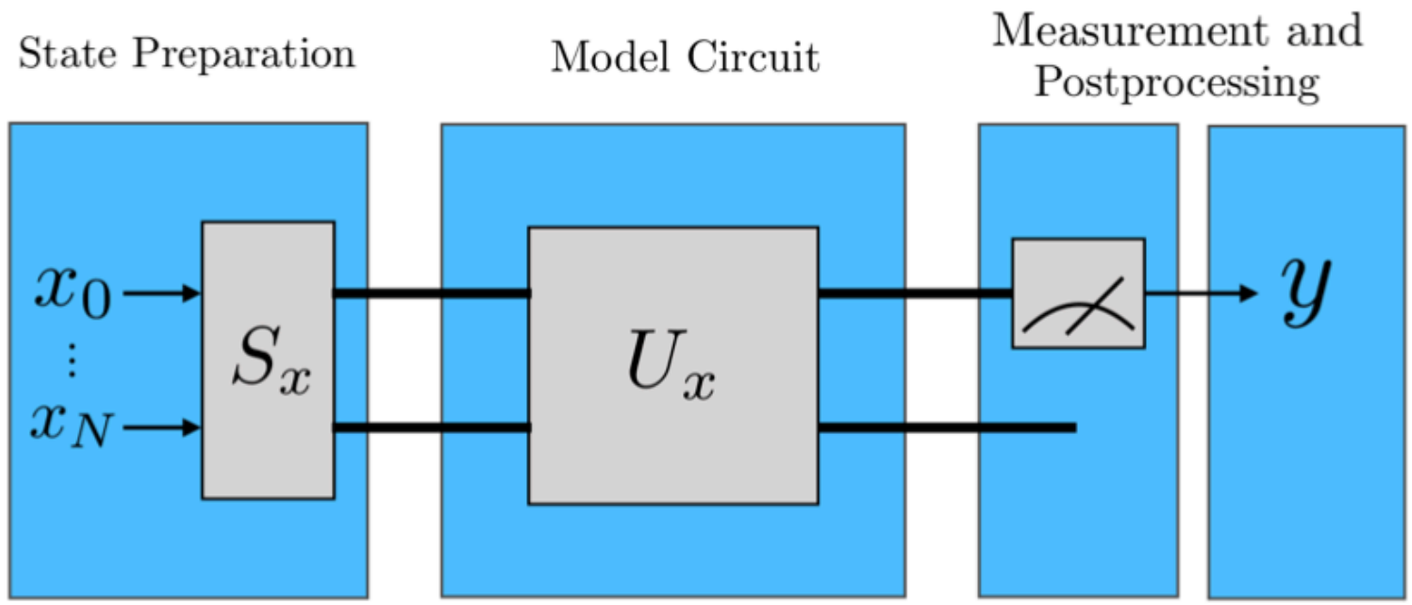
Quantum Machine Learning with a Variational Quantum Circuit

[McClean et al '16]

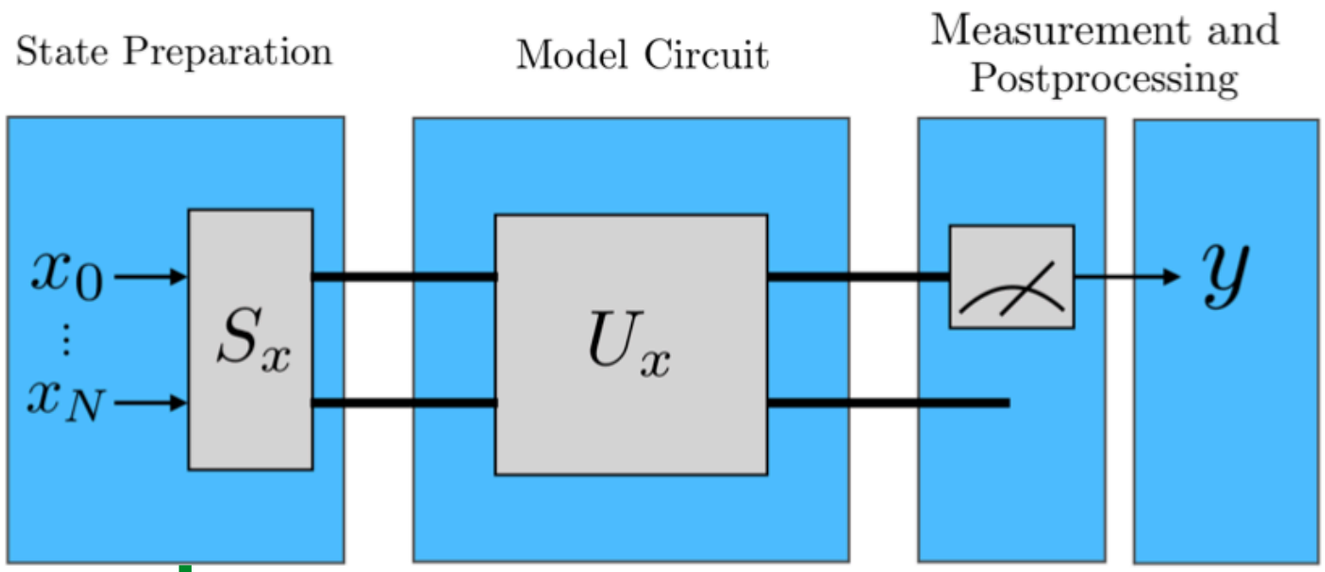
[Farhi, Neven '18]

[Schuld et al '20]

[Blance, MS '20]



Quantum Machine Learning with a Variational Quantum Circuit



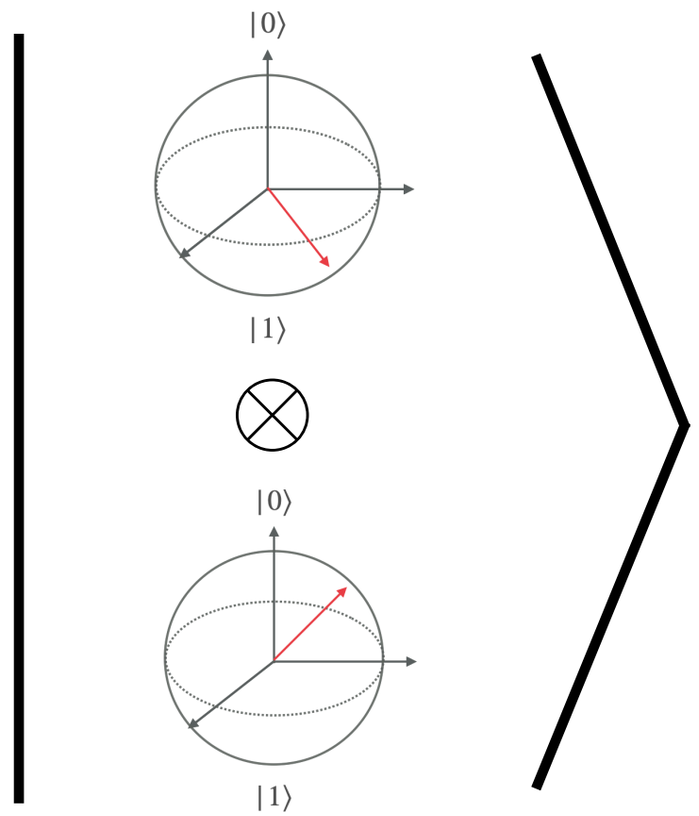
state preparation

n corresponds to # features

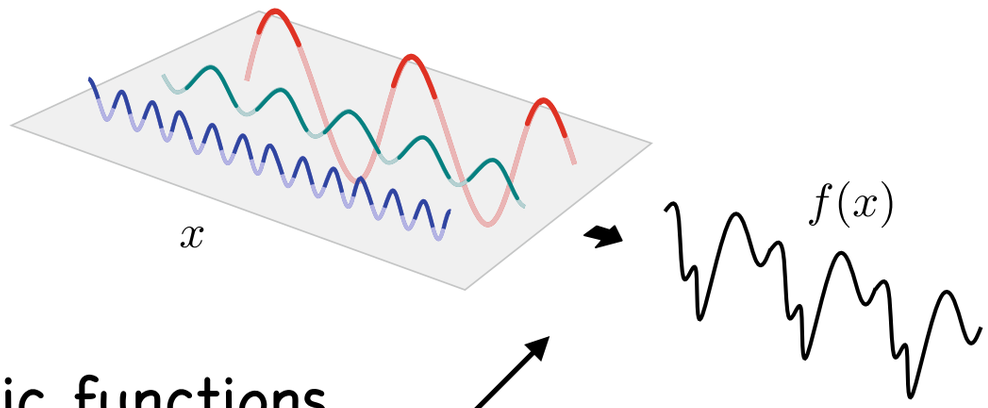
$$x \mapsto S_x|\phi\rangle = S_x|0\rangle^{\otimes n} = |x\rangle$$

e.g. angle encoding

$$|x\rangle = \bigotimes_{i=1}^{n} \cos(x_i)|0\rangle + \sin(x_i)|1\rangle$$



Expressibility of model and encoding



- Most encodings result in sum of trigonometric functions, e.g. angle encoding, time evolution encoding

- Fourier series is universal approximator, but problem is that for many encoding strategies quantum models are linear combinations of functions composed of very few frequencies

- If encoding gates are not rich enough -> model limited irrespective of width or depth in variational circuit

  ➡ Encoding controls expressivity of model
  (circuit expressivity != model expressivity)

  ➡ Fourier analysis for VQC (mindful about pre-scaling)

  ➡ Can hint on where VQC models are particularly useful

Specifically, for many (angle, time-evolution,...) encoding gates are of form

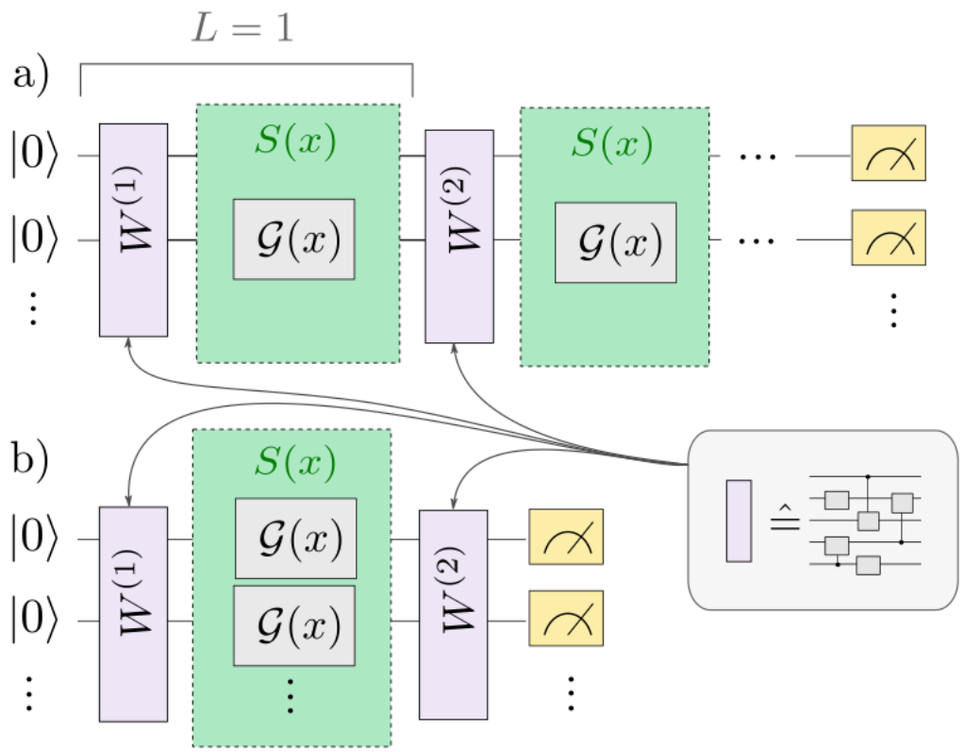
$$S(x) = e^{-ixG}$$

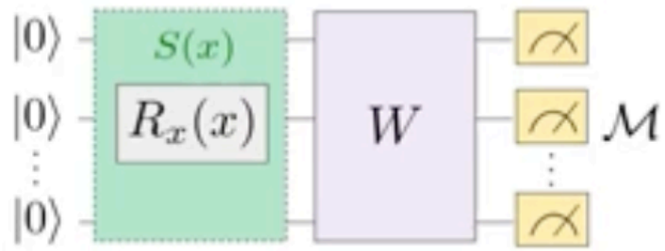
For Pauli operators output can be expressed as Fourier series

(universal approximator, dependent on number of frequencies)

encoding multiple times

encoding in parallel





$$f_\theta(x) = \langle \mathcal{M} \rangle_{x,\theta} = A + B\cos(x) - C\sin(x)$$

A, B, C coefficients from parametrised circuit W

trigonometric structure from data encoding

See 2008.08605
1907.02085

Simple example for 1-qubit system and 1 encoding operation:

$$f_\theta(x) = \langle 0|U^\dagger(x,\theta)\sigma_z U(x,\theta)|0\rangle \qquad x \in \mathcal{X} = \mathbb{R}$$

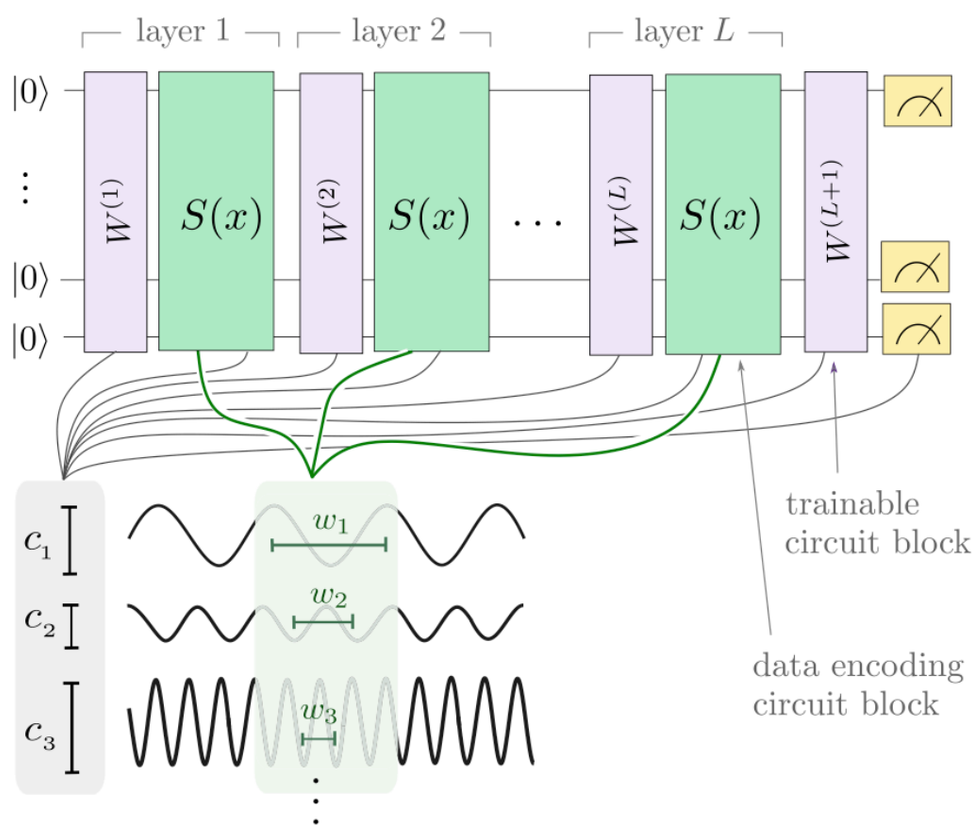
take $U(x,\theta) = W(\theta)R_x(x)$ $\qquad W(\theta) = \mathrm{Rot}(\theta_1,\theta_2,\theta_3)$ $\qquad R_x(x) = e^{-i\frac{x}{2}\sigma_x}$

Model output

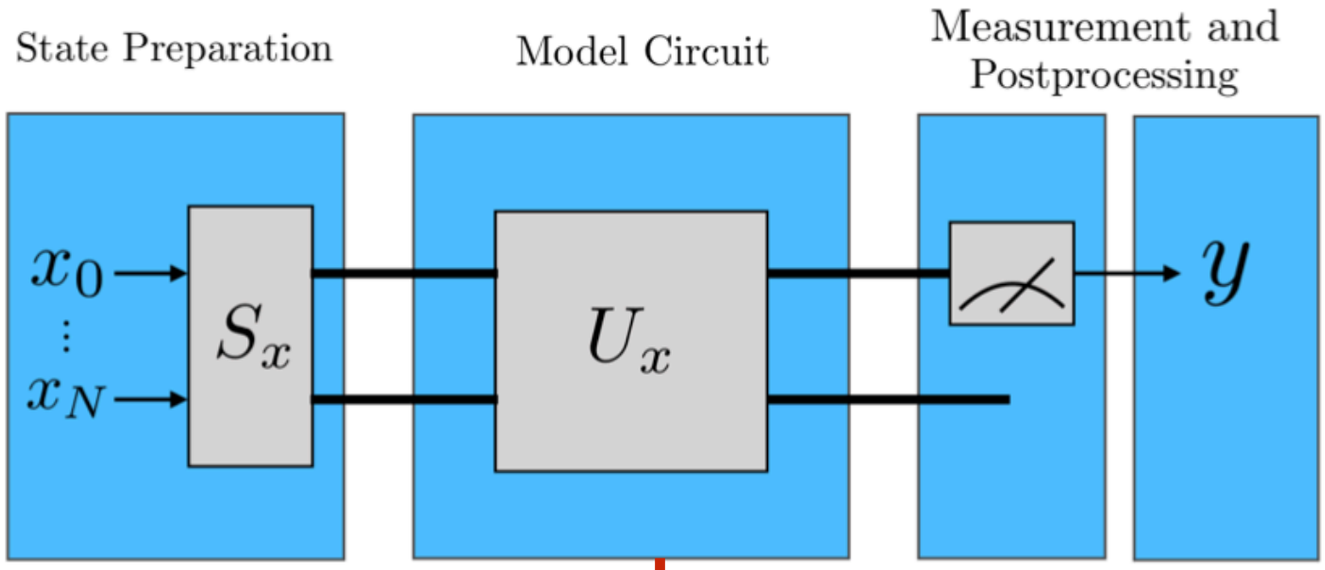
$$f_\theta(x) = \sum_{n=-1}^{1} c_n e^{inx} = c_{-1}e^{-ix} + c_0 + c_1 e^{ix} = c_0 + 2\mathrm{Re}(c_1)\cos(x) + 2\mathrm{Im}(c_1)\sin(x)$$

- The encoding of data, using e.g. angle or time-evolution encoding, results in a trigonometric structure of the model's output.

- Complex enough (potentially repeated) encoding results in modes with different frequencies -> **universal approximator**



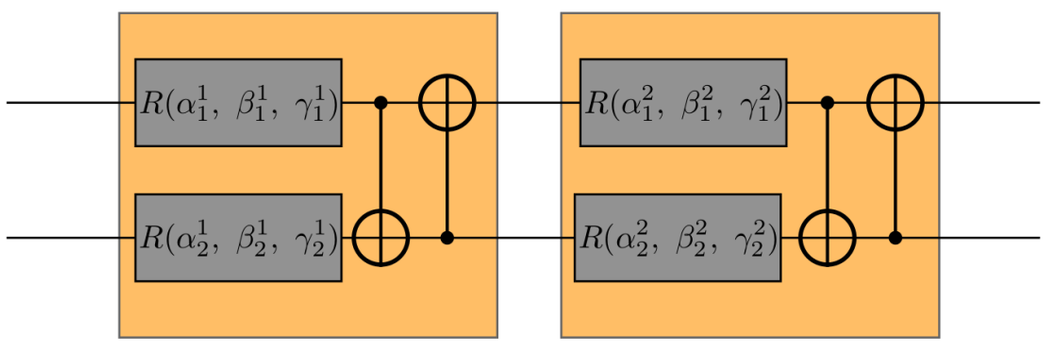
Quantum Machine Learning with a Variational Quantum Circuit



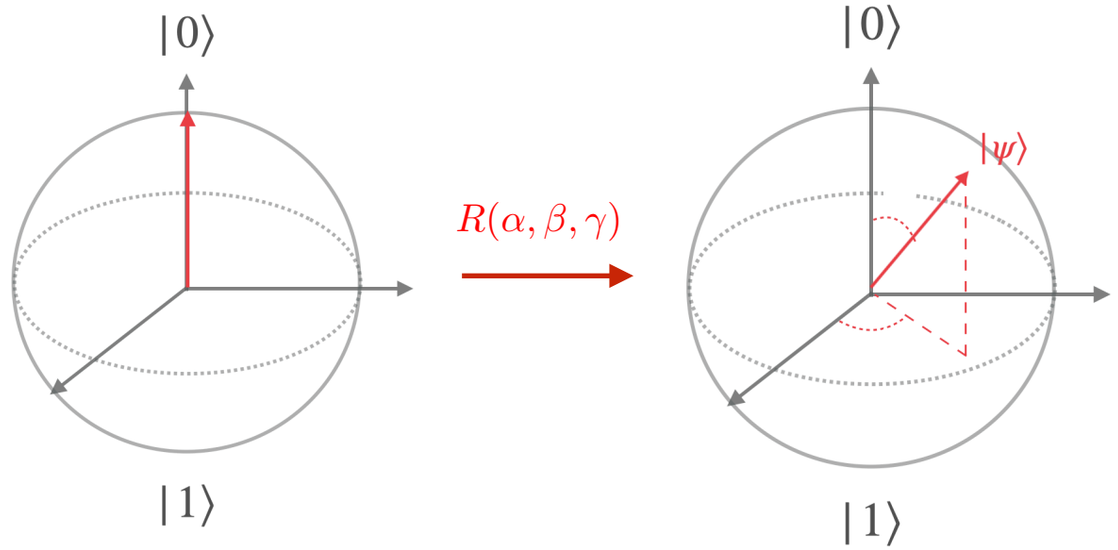
$$|\psi\rangle = U(w)|x\rangle \quad \text{with} \quad U(w) = U_{l_{\max}}(w_{l_{\max}})...U_l(w_l)...U_1(w_1)$$

model circuit — trainable parameters — prepared state

2-layer Variational Quantum Circuit

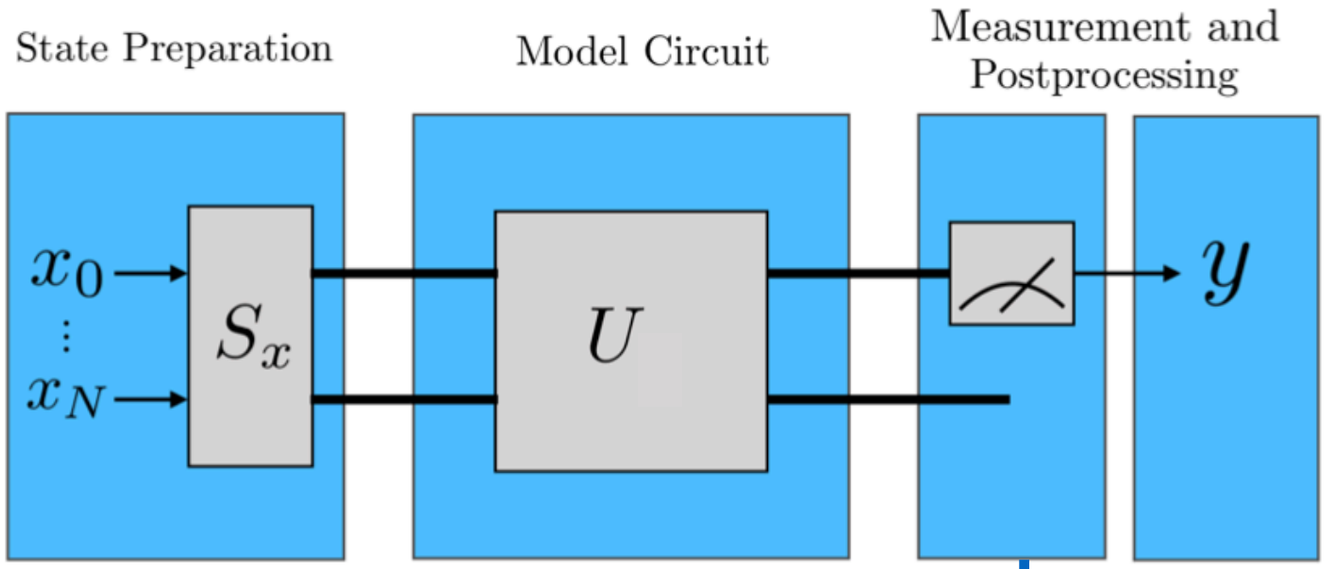


U_1 U_2



→ Rotation + CNOT -> Entanglement

Quantum Machine Learning with a Variational Quantum Circuit



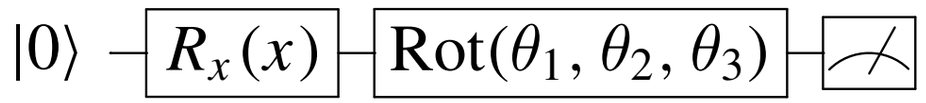
- Entangled state shares information across qubits
- Evaluate expectation value of qubits to construct loss

  for supervised S vs B classification one qubit sufficient

$$\mathbb{E}(\sigma_z) = \langle 0|S_x(x)^\dagger U(w)^\dagger \hat{O} U(w) S_x(x)|0\rangle = \pi(w,x) \quad \text{for} \quad \hat{O} = \sigma_z \otimes \mathbb{I}^{\otimes(n-1)}$$

- Quantum network output: $f(w,b,x) = \pi(w,x) + b$
- Changing operator and loss => VQE, VQT, ... (simulate QFT)

Simple example:



gives the model output

$$f_\theta(x) = \langle 0|R_x(x)^\dagger \mathrm{Rot}(\theta_1, \theta_2, \theta_3)^\dagger \sigma_z \mathrm{Rot}(\theta_1, \theta_2, \theta_3) R_x(x)|0\rangle$$

data encoding

$$|\phi(x)\rangle = R_x(x)|0\rangle = \begin{pmatrix} \cos(\frac{x}{2}) & -i\sin(\frac{x}{2}) \\ -i\sin(\frac{x}{2}) & \cos(\frac{x}{2}) \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \cos(\frac{x}{2}) \\ -i\sin(\frac{x}{2}) \end{pmatrix}$$

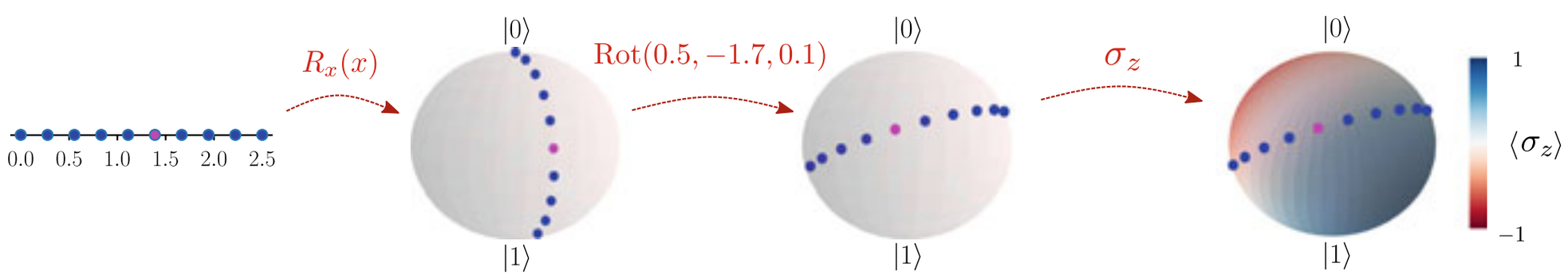
parametrised rotation

$$\begin{aligned} |\psi(x,\theta)\rangle &= \mathrm{Rot}(\theta_1, \theta_2, \theta_3)|\phi(x)\rangle \\ &= \begin{pmatrix} e^{i(-\frac{\theta_1}{2} - \frac{\theta_3}{2})} \cos(\frac{\theta_2}{2}) \cos(\frac{x}{2}) + i e^{i(-\frac{\theta_1}{2} + \frac{\theta_3}{2})} \sin(\frac{\theta_2}{2}) \sin(\frac{x}{2}) \\ e^{i(\frac{\theta_1}{2} - \frac{\theta_3}{2})} \sin(\frac{\theta_2}{2}) \cos(\frac{x}{2}) - i e^{i(\frac{\theta_1}{2} + \frac{\theta_3}{2})} \cos(\frac{\theta_2}{2}) \sin(\frac{x}{2}) \end{pmatrix} \end{aligned}$$

model output

$$f_\theta(x) = \langle \psi(x,\theta)|\sigma_z|\psi(x,\theta)\rangle = \cos(\theta_2)\cos(x) - \sin(\theta_1)\sin(\theta_2)\sin(x)$$

What happens on the Bloch-Sphere



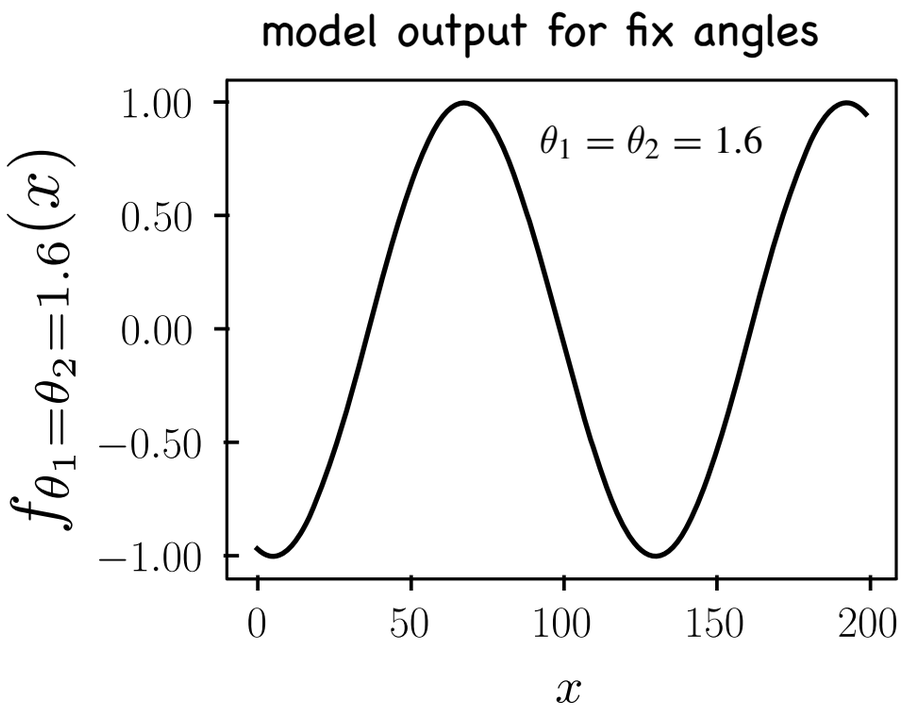
Classifier from simple example:

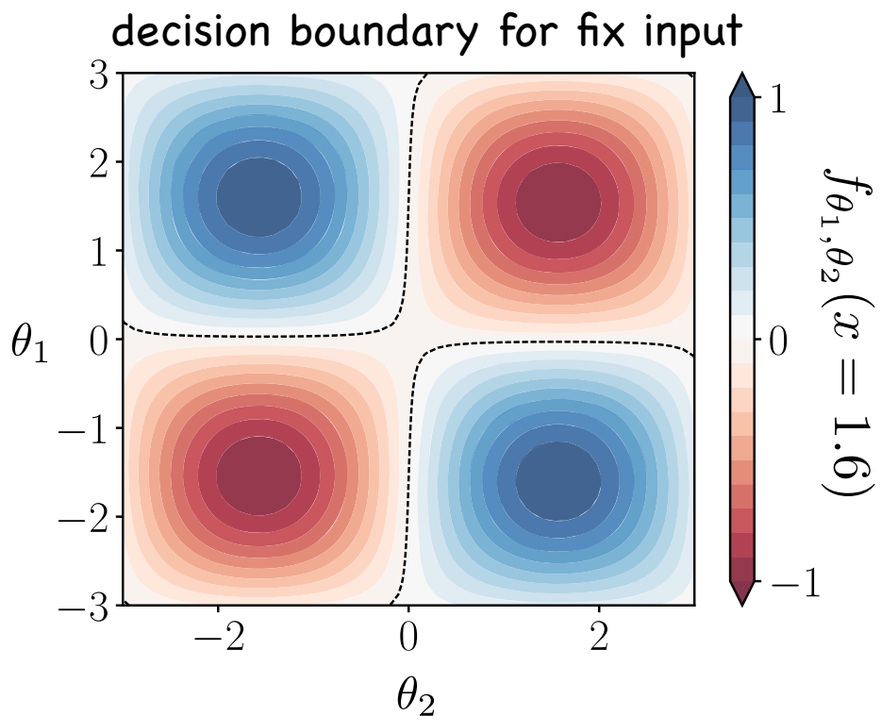
for binary classifier define e.g.

$$y = \begin{cases} 1 \text{ if } f_\theta(x) > 0 \\ -1 \text{ else} \end{cases}$$

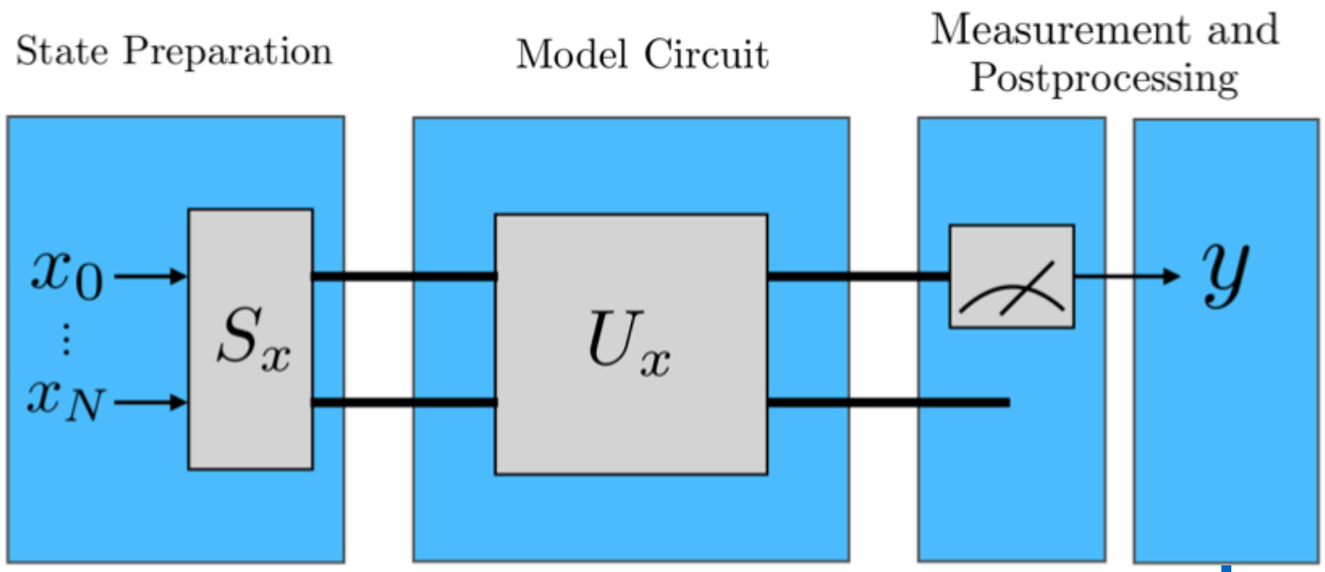
for probabilistic classifier (density estimator)

$$p(1) = \frac{f_\theta(x) + 1}{2}, \qquad p_0 = 1 - p_1$$



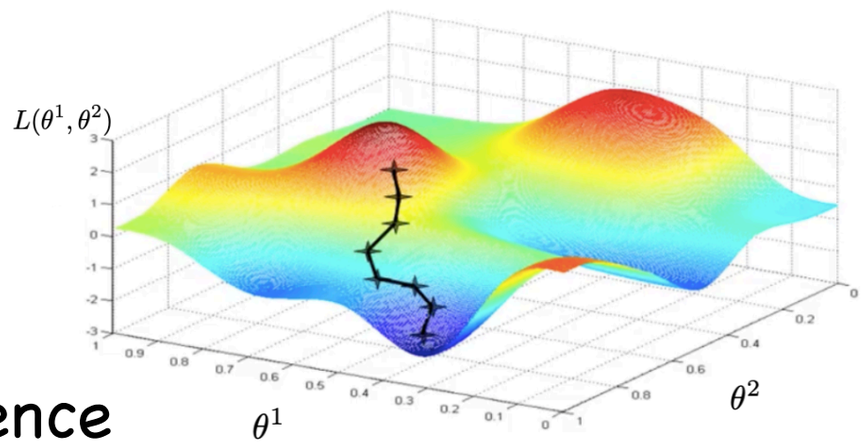


Quantum Machine Learning with a Variational Quantum Circuit



- Hybrid approach (QC to calculate exp. value, CC to optimise U operator)
- Loss function $L = \frac{1}{n}\sum_{i=1}^{n}\left[y_i^{\text{truth}} - f(w,b,x_i)\right]^2$

label (signal, bkg), supervised learning



- Quantum gradient descent - for fast convergence

Fubiny-Study metric underlies geometric structure of VQC parameter space: $\theta_{t+1} = \theta_t - \eta g^{+}\nabla L(\theta)$

[Cheng '10]
[Blance, MS '20]
[Abbas et al '20]

Training VQC

- Backpropagation to adjust trainable parameters - as for classical NN
- Calculation of derivatives of cost function

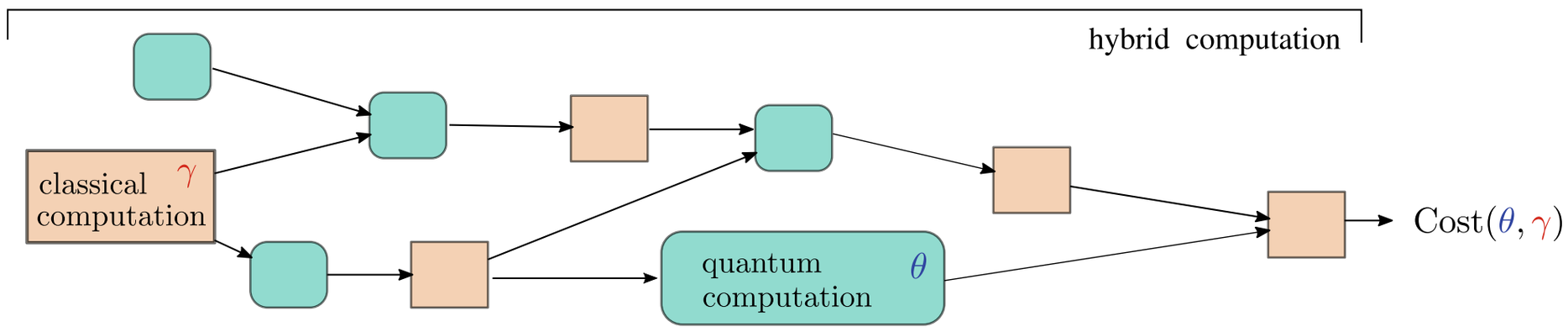
$$\partial_\mu L(\theta) = \frac{\partial L}{\partial f_\theta}\frac{\partial f_\theta}{\partial \mu}$$

model output result of quantum computation

classical calculation

how to calculate $\frac{\partial f_\theta}{\partial \mu}$?

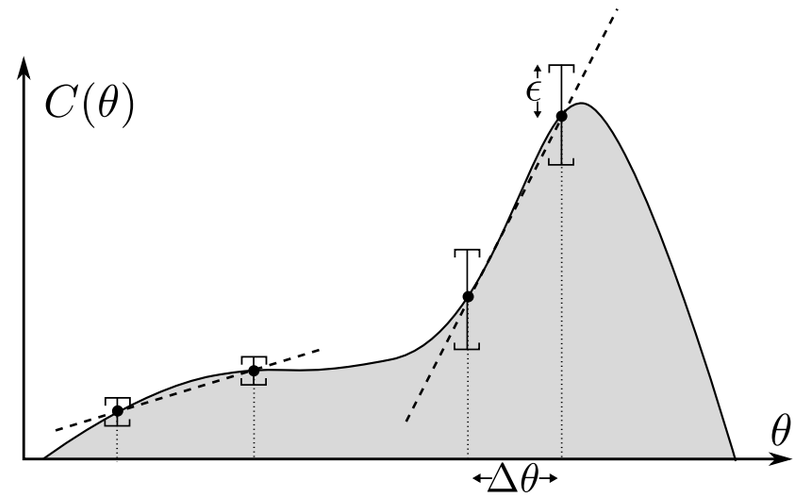
- If such partial derivative is calculable, hybrid approach straightforward



Gradients for VQC outputs

- Finite difference method (FDM) always possible

$$\frac{\partial f_\theta}{\partial \mu} \approx \frac{f_\theta - f_{\theta+\Delta\theta}}{\|\Delta\theta\|}$$



- The flatter the gradient, the more precisely the function has to be evaluated

⟶ FDM increasingly problematic for small gradients

- Another problem - single gate quantum circuit $\mathcal{G}(\mu)$ with $f_\mu = \langle\psi|\mathcal{G}^\dagger(\mu)\mathcal{B}\mathcal{G}(\mu)|\psi\rangle$

$$\partial_\mu\langle\psi|\mathcal{G}^\dagger(\mu)\mathcal{B}\mathcal{G}(\mu)|\psi\rangle = \langle\psi|\mathcal{G}^\dagger\mathcal{B}(\partial_\mu\mathcal{G})|\psi\rangle + \langle\psi|(\partial_\mu\mathcal{G})^\dagger\mathcal{B}\mathcal{G}|\psi\rangle$$

not clear if unitary

each term not quantum expectation value

⟶ Can use shift rule and apply entire model twice!

Parameter-shift Rules

We can calculate the partial derivative for quantum exp. value $f_\mu = \langle \mathcal{M} \rangle_\mu$ as

$$\partial_\mu f_\mu = \sum_i a_i \, f_{\mu + s_i}$$

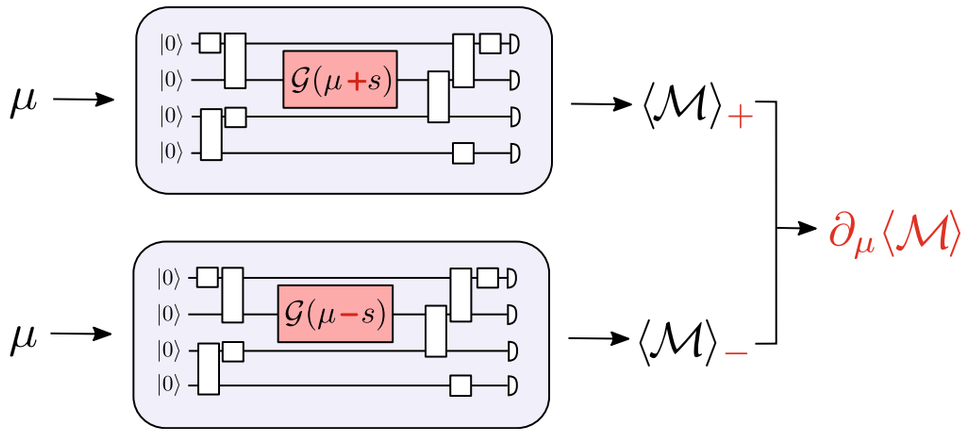
With real $\{a_i\}$ and $\{s_i\}$ and the shift not necessarily being small. Gradient is exact!

⟶ finite difference computes estimation of approximate gradient

⟶ parameter shift computes estimation of the analytic gradient (of exp val)

All unitaries of Pauli rotations and their tensor products can be expressed as

$$\mathcal{G}(\mu) = e^{-i\mu G} = \cos(\mu)\mathbb{1} - i\,\sin(\mu) G$$



$$\partial_\mu f_\mu = \frac{1}{2\sin(s)} \left(\langle \psi | \mathcal{G}^\dagger(\mu + s) B \mathcal{G}(\mu + s) | \psi \rangle - \langle \psi | \mathcal{G}^\dagger(\mu - s) \mathcal{B} \mathcal{G}(\mu - s) | \psi \rangle \right) = \frac{1}{2\sin(s)} \left(f_{\mu+s} - f_{\mu-s} \right)$$

⟶ However computationally costly (evaluate circuit twice) s often chosen $\pi/2$

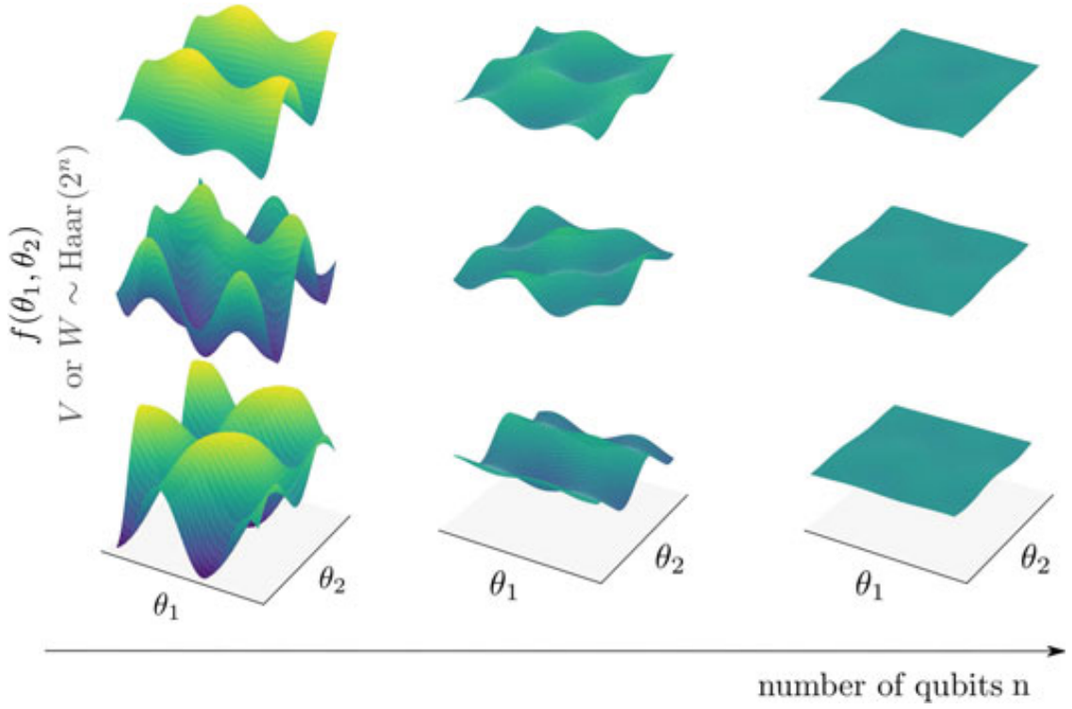
Barren Plateaus

- Area in loss landscape where gradients are close to zero
- Optimisation is slow and expensive, requiring high accuracy in evaluating gradient to avoid random walking

⟶ Barren Plateaus often arise if quantum model is overly expressive and Hilbert spaces are large

⟶ Individual gradient steps in exponentially large parameter and Hilbert space becomes less relevant

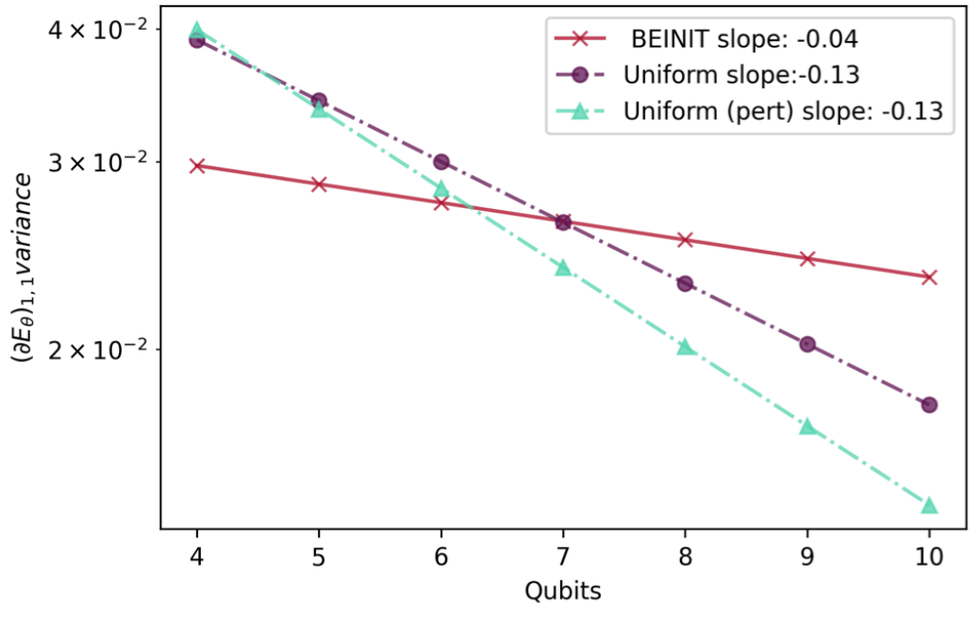
⟶ Important task for efficient learning is the choice of model, i.e. **as expressive as necessary while as small as possible**

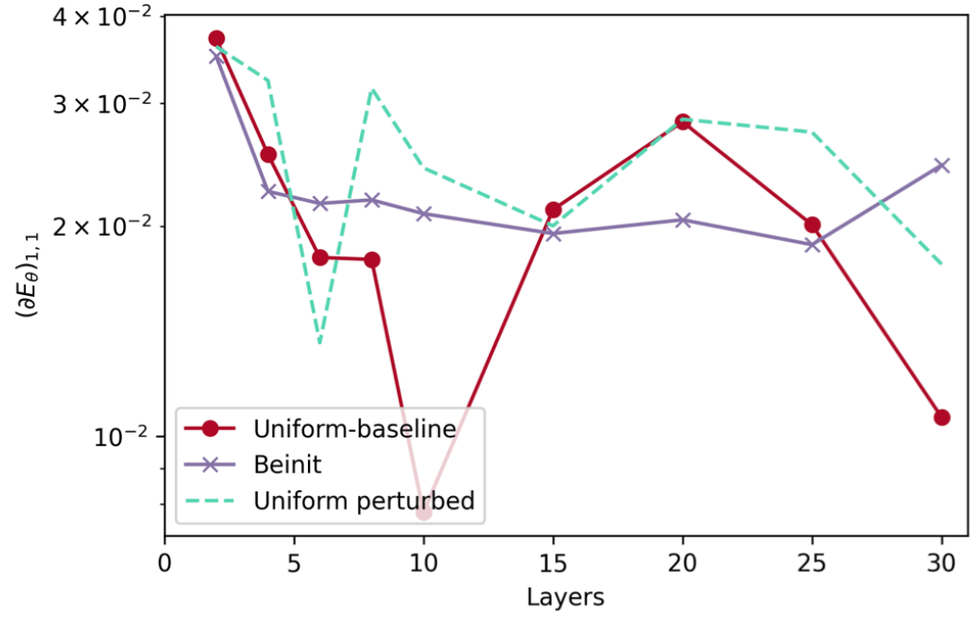


Some ways to avoid Barren plateaus

- BEINIT Avoid Barren Plateaus in Variational Quantum Algorithms
  - ➡ A strategy that initialises the parameters of a unitary gate by drawing from a beta distribution. The hyperparameters of the beta distribution are estimated from the data.
  - ➡ To further prevent barren plateau during training, a perturbation is added at every gradient descent step.
  - ➡ This framework significantly reduces the possibility of a complex quantum neural network getting stuck in a barren plateau.

[Kulshrestha, Safro '22]





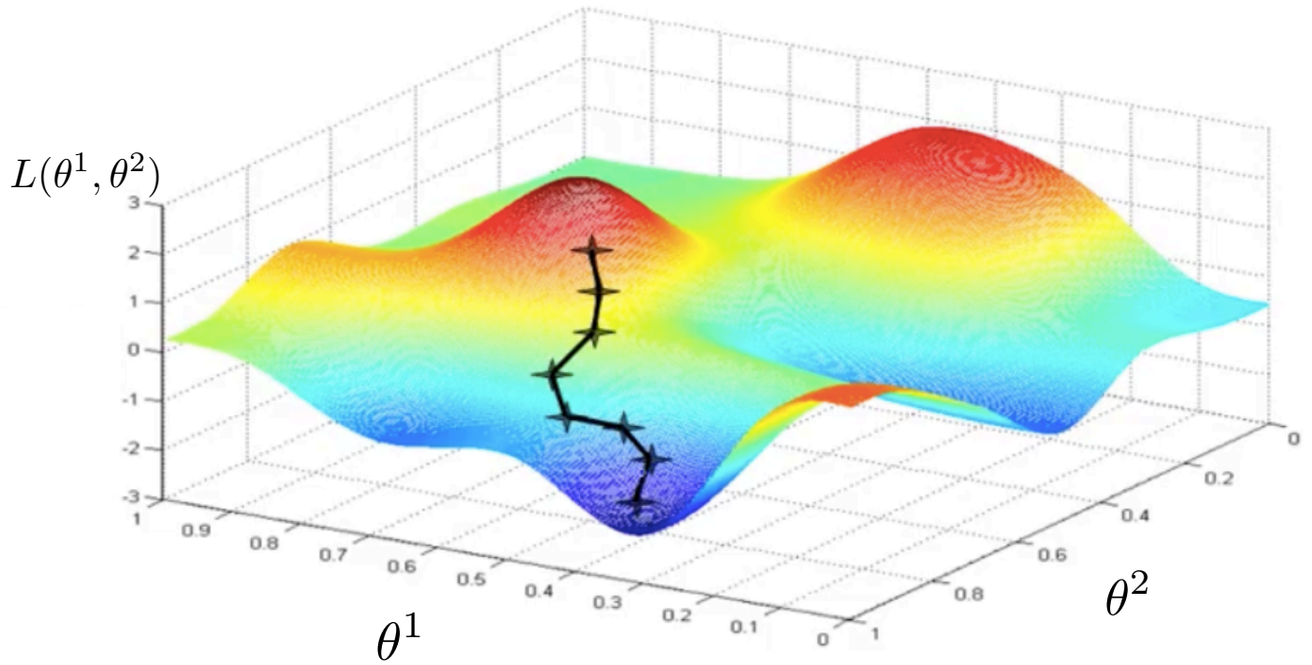
- Avoiding Barren Plateaus using classical shadows

[Sack, Medina, Michailidis et al '22]

Optimising the loss landscape

<u>classical gradient descent (GD):</u>

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta)$$



<u>quantum gradient descent (QDC):</u>

Fisher Information Matrix F promotes
gradient descent to natural gradient descent (Riemannian geometry):

$$\theta_{t+1} = \theta_t - \eta F^{-1} \nabla L(\theta)$$

[Stokes, Izaac, Killoran, Carleo '20] [Blance, MS '20]

Fubiny-Study metric underlies
geometric structure of VQC
parameter space (complex
projective Hilbert Spaces):

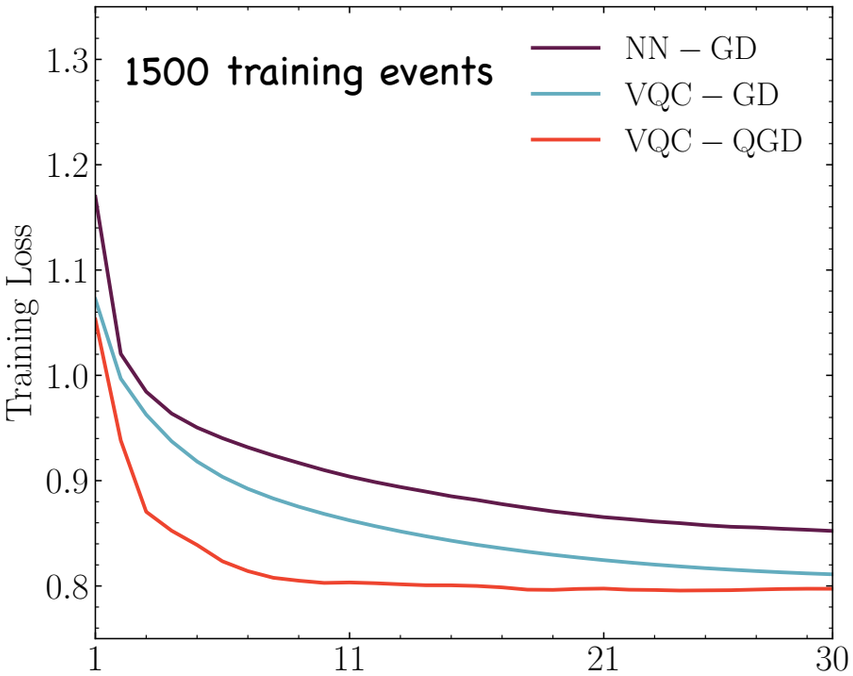
$$\theta_{t+1} = \theta_t - \eta g^{+} \nabla L(\theta)$$

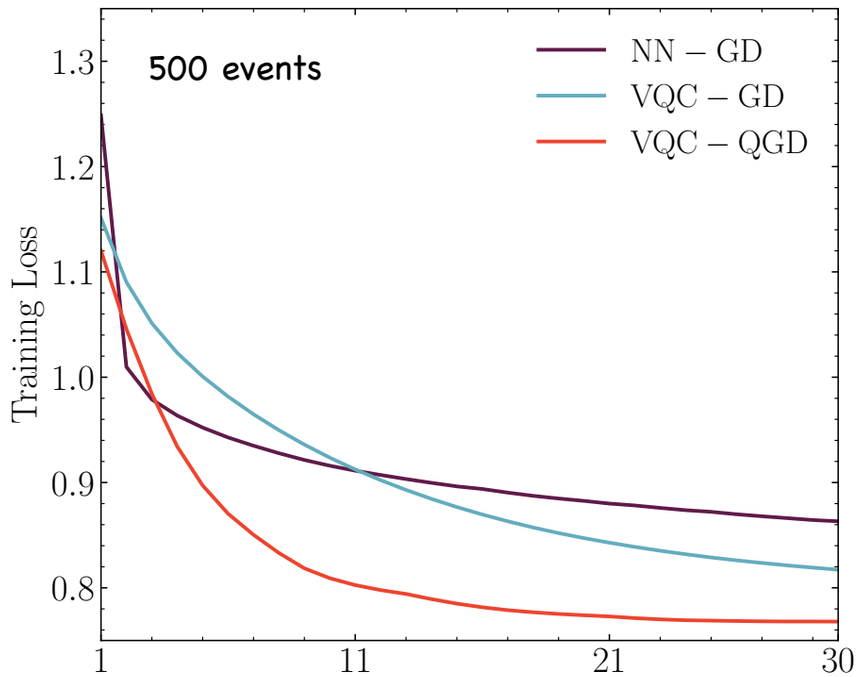
VQC parameters

weights $\theta^w_{t+1} = \theta^w_t - \eta g^{+} \nabla^w L(\theta)$,

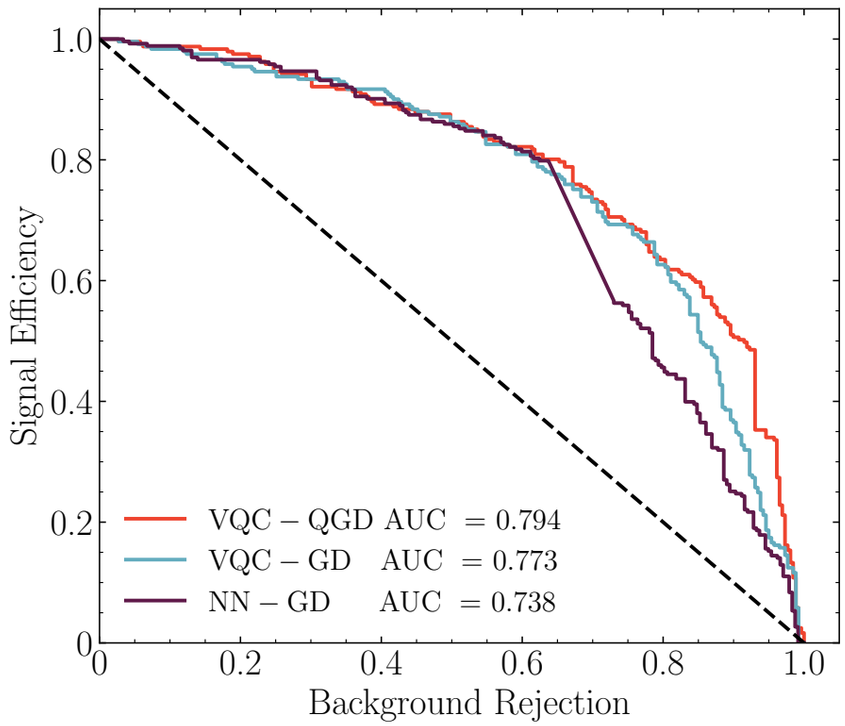
bias $\theta^b_{t+1} = \theta^b_t - \eta \nabla^b L(\theta)$,

Gate quantum machine learning in action





[Blance, MS '20]



QC device vs simulator

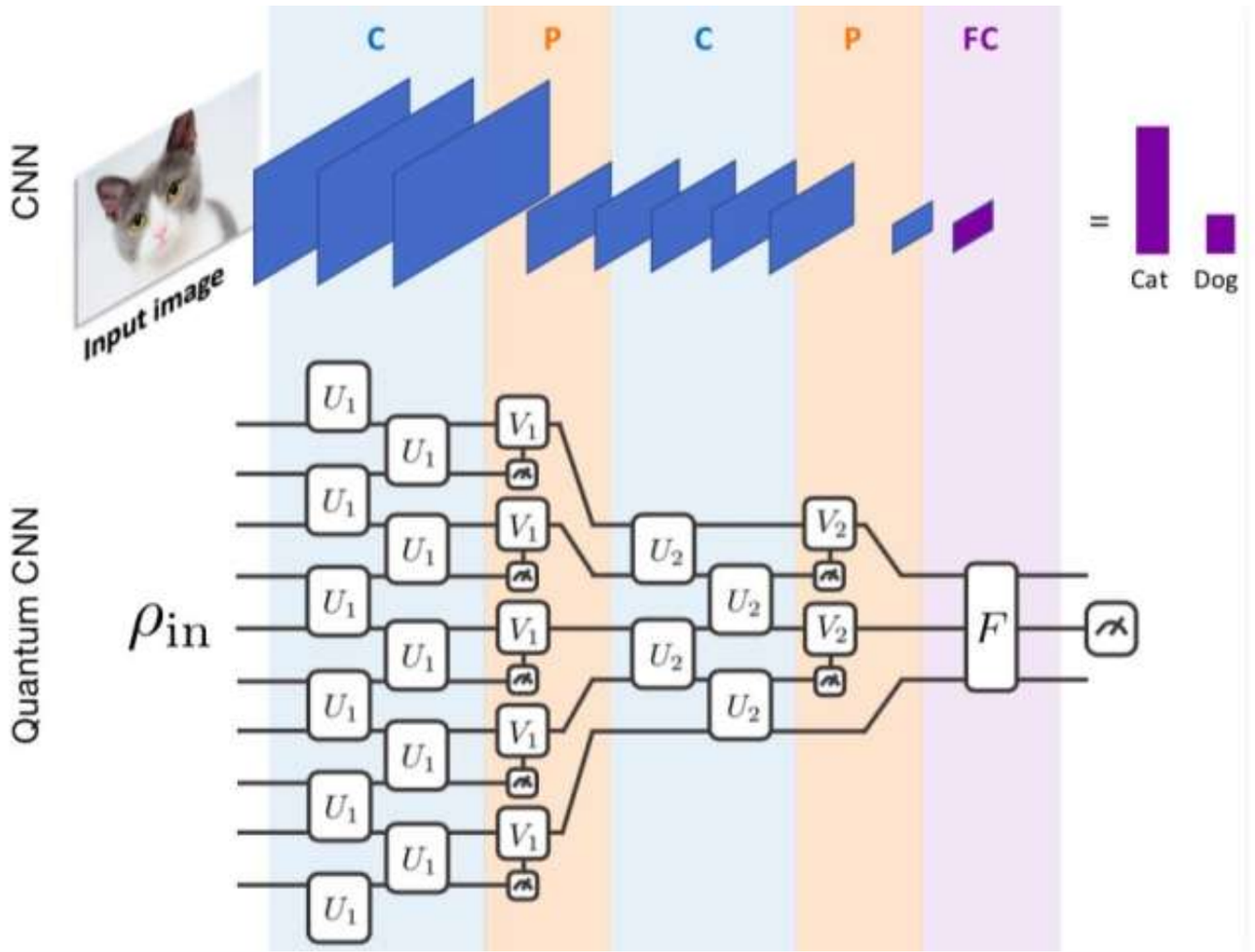
Device	Accuracy (%)
PennyLane default.qubit	72.6
ibmq_qasm_simulator	72.6
ibmqx2	71.4

- Applied to $pp \to t\bar{t}$ vs $pp \to Z' \to t\bar{t}$ lept. top dec for 2d feature space only p_{T,b_1} and $\not{E}_T$

Quantum Convolutional Neural Network

- Convolutional layers work by sweeping across the input array and applying different filters (often 2x2 or 3x3 matrices) block by block.
  Used to detect specific features of the image wherever they might appear.

- Pooling layers used to downsample results of these convolutions to extract most relevant features and reduce the size of the data.
  Common pooling methods involve replacing blocks of the data with their maximum or average values.



conv layer pooling layer Fully connected layer

- QCNN uses only O(log(N)) variational parameters for input size of N qubits

[Cong, Choi, Lukin '19]

Convolutional Layer

- In the QCNN, each layer contains parametrized circuits, meaning we alter our output result by adjusting the parameters of each layer. When training our QCNN, it is these parameters that are adjusted to reduce the loss function of our QCNN.

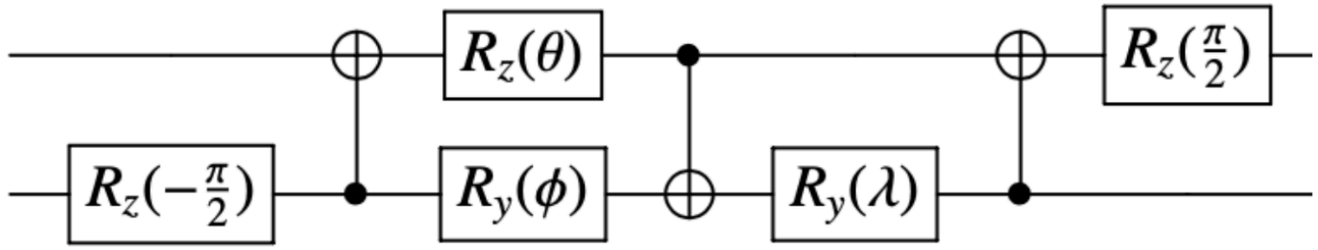
- Assuming 2 qubits, a general Unitary operator for a conv layer could be written as

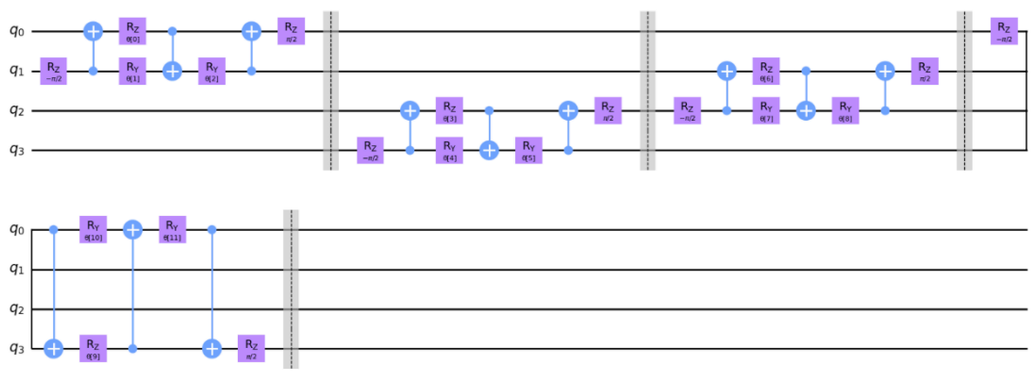
$$U = (A_1 \otimes A_2) \cdot N(\alpha, \beta, \gamma) \cdot (A_3 \otimes A_4) \qquad \text{with} \qquad N(\alpha, \beta, \gamma) = exp(i[\alpha\sigma_x\sigma_x + \beta\sigma_y\sigma_y + \gamma\sigma_z\sigma_z]),$$
$$A_j \in \mathrm{SU}(2)$$

each U has 15 trainable parameters ⟶ long training times -> Simplify ansatz:

[quant-ph/0308006]

Connect all adjacent qubits with this ansatz





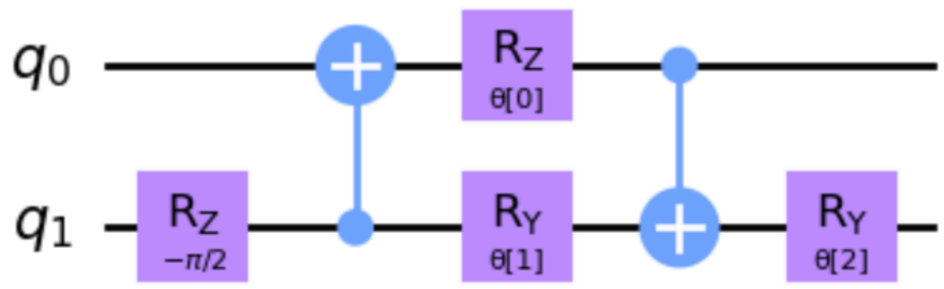
eg. for 4 qubits, like that

Pooling Layer

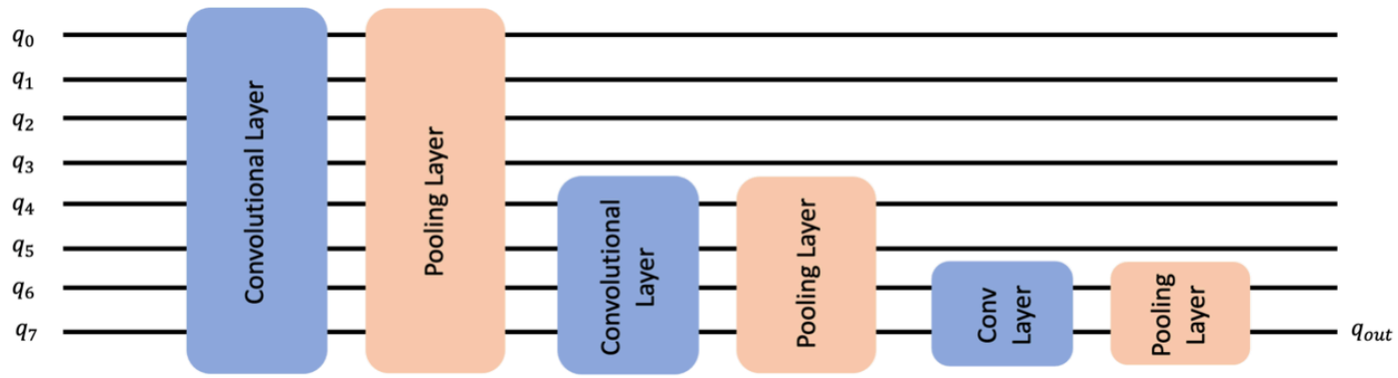
- The purpose of a pooling layer is to reduce the dimensions of our Quantum Circuit, i.e. reduce the number of qubits in our circuit, while retaining as much information as possible from previously learned data.

- Two options:
  1) ignore some of the qubits after entangling them with adjacent qubits
  2) perform a measurement of qubits and use classical feedback loop to process outcome

- Concretely for 1). Apply this circuit and ignore qubit q0 in everything that follows.



- output can be processes via FC classical, quantum network or directly into loss



Some results

Input to QCNN: ground state wave function for Hamiltonian

$$\hat{H} = -J\sum_{i=1}^{n-2}(\hat{Z}_i\hat{X}_{i+1}\hat{Z}_{i+2}) - h_1\sum_{i=1}^{n}(\hat{X}_i) - h_2\sum_{i=1}^{n-1}(\hat{X}_i\hat{X}_{i+1})$$

with couplings (J, h1, h2)

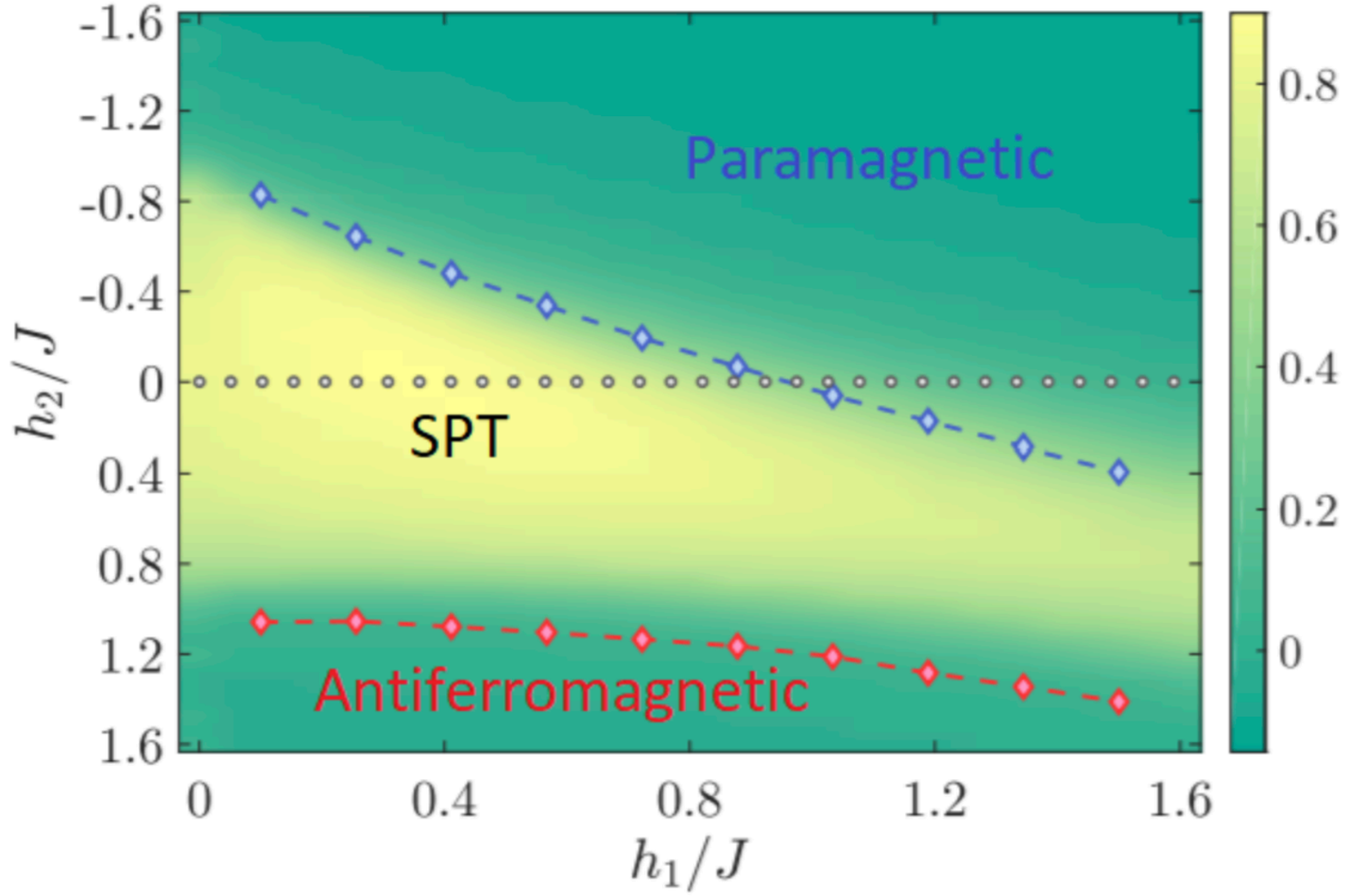
[Cong, Choi, Lukin '19]

[Nagano, et al '23]

Grey dots, training data set

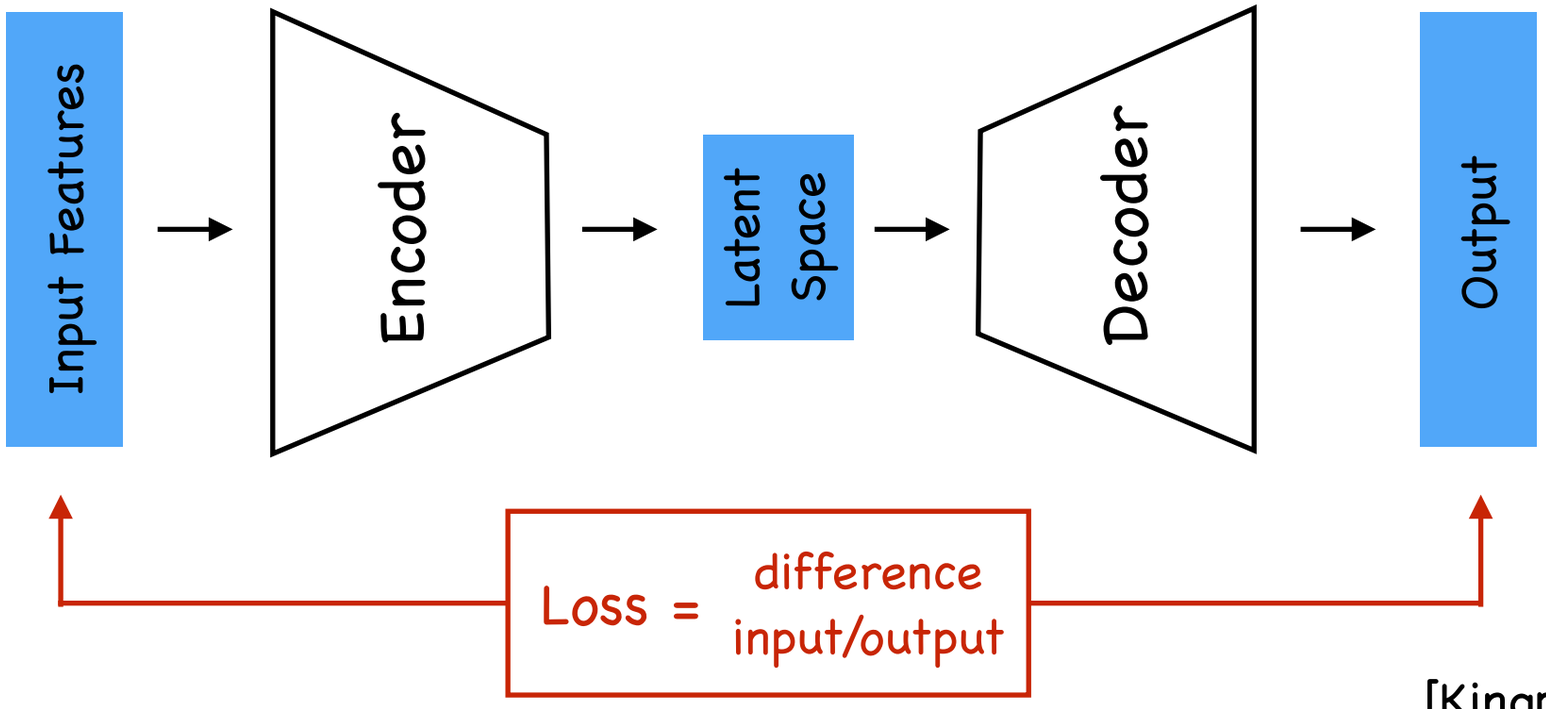
Red and blue dots are true boundaries as calculated by DMRG

Yellow and green shaded area is NN output for the different phases



Autoencoder for unsupervised learning

Most popular NN-based anomaly detection method

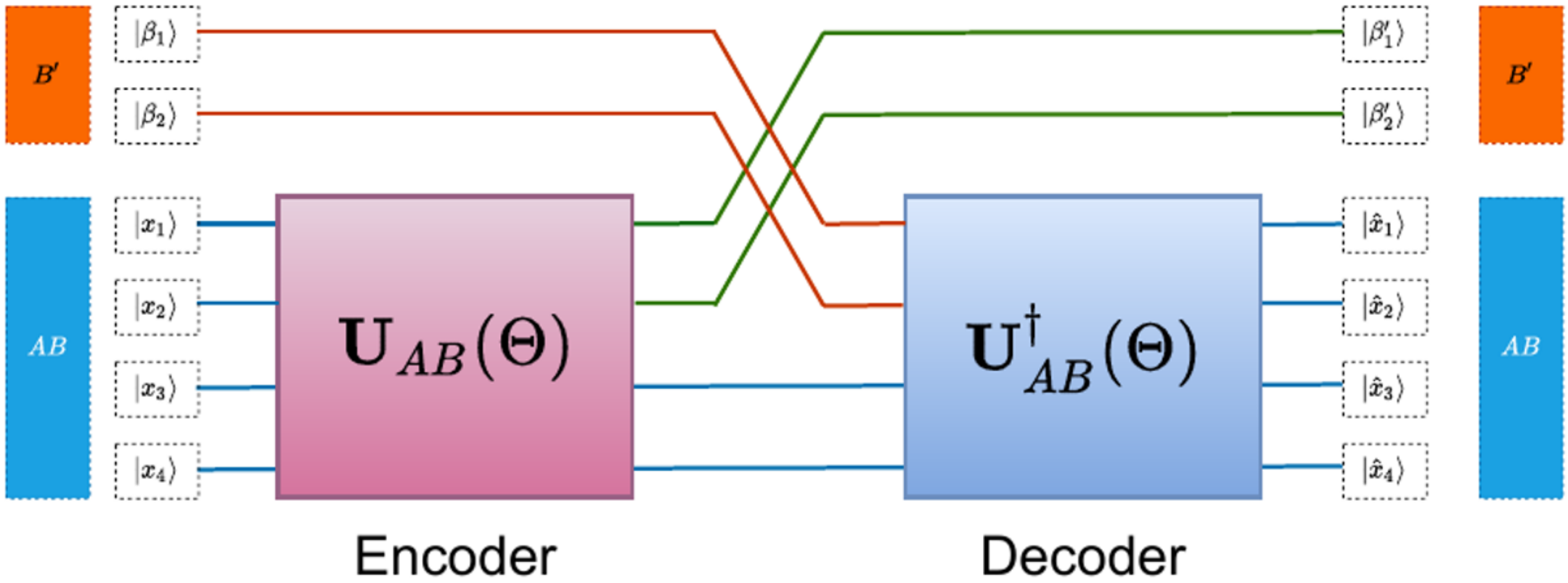


[Kingma, Welling '13]

- in first step input is encoded into information bottleneck
- between input/output layer and bottleneck can be several hidden layers (conv./deep NNs) -> highly non-linear
- after bottleneck decoding step
- Reconstructed output is then compared with input via loss-function (often MSE)
- NN is trained such that input and output high degree of similarity

Unsupervised learning with quantum-gate Autoencoder

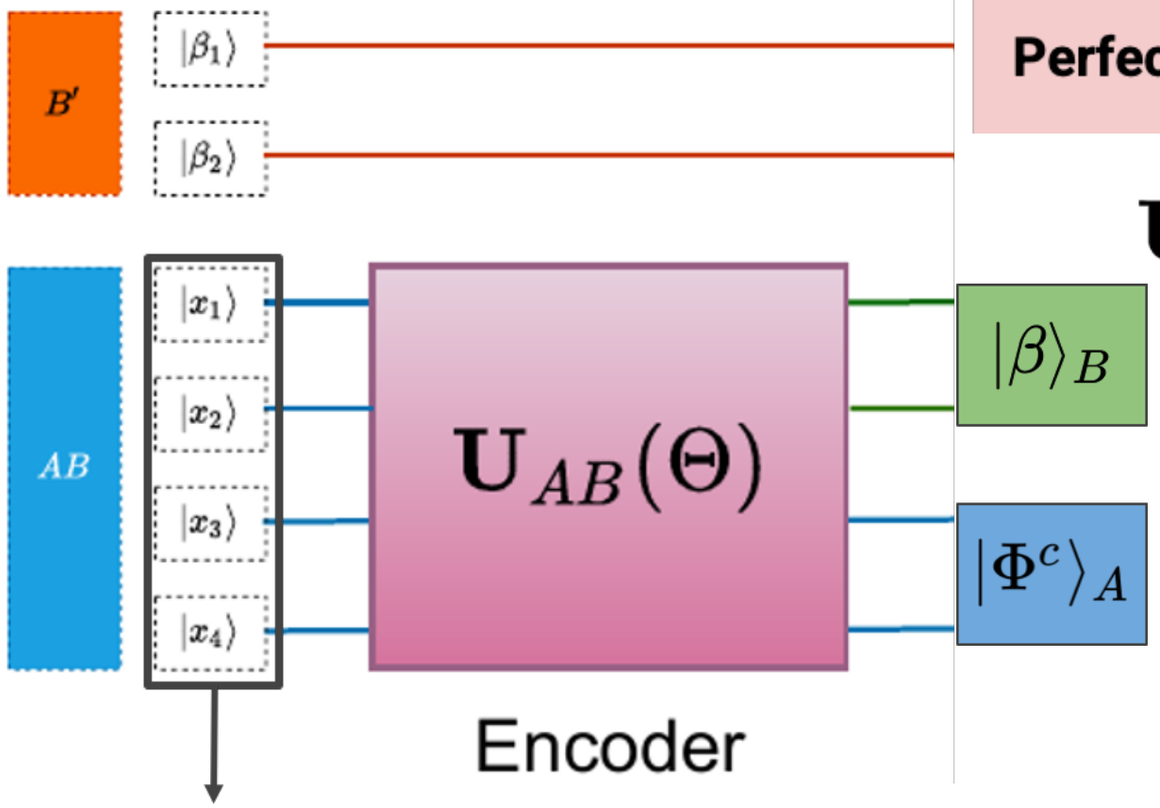
[Ngairangbam, MS, Takeuchi '21]



Quantum unitary transformations = probability conserving!

Induce ***information bottleneck*** by **discarding states of B system** after encoding, and **replacing with reference states B'** with no connection with the encoder.

Unsupervised learning with quantum-gate Autoencoder



Perfect reconstruction possible if

$$\mathbf{U}_{AB}|\Phi\rangle_{AB} = |\Phi^c\rangle_A \otimes |\beta\rangle_B$$

Contains all information of input

Train by tuning the unitary so that the output of the B subsystem is same as the reference state in B'

$$|\Phi\rangle_{AB}$$

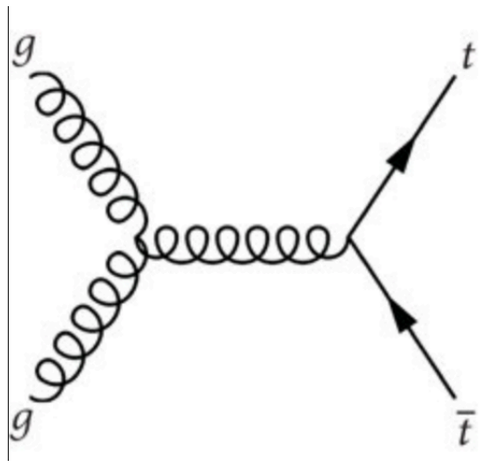
$$|\chi\rangle_{AB} \in \mathcal{H}_A \otimes \mathcal{H}_B$$

$$\mathbf{U}_{AB}|\Phi\rangle_{AB} = |\chi\rangle_{AB}$$

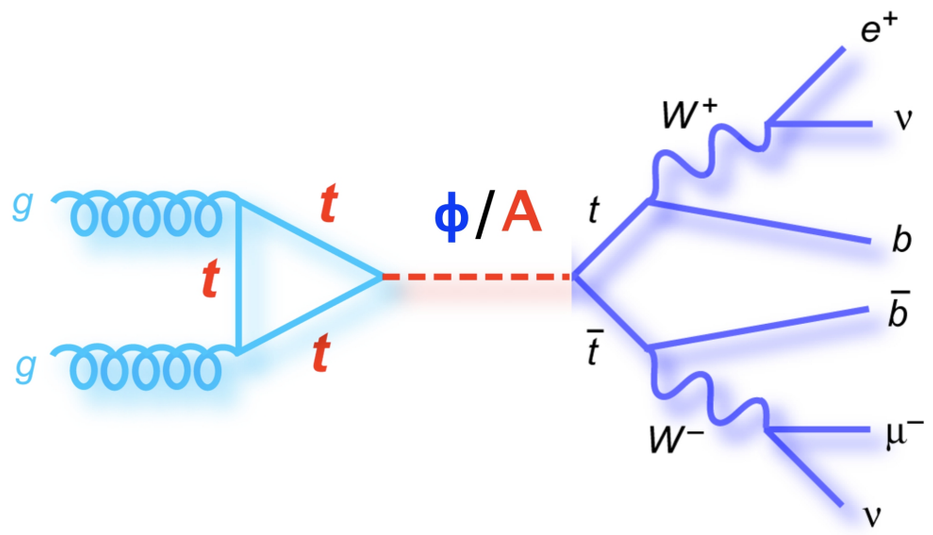
Quantum similarity measured via fidelity

$$F(|\phi\rangle, |\psi\rangle) = F(|\psi\rangle, |\phi\rangle) = |\langle\phi|\psi\rangle|^2$$

- background



signal



$$t\,\bar{t} \rightarrow b\,\bar{b}\,W^+\,W^- \rightarrow b\,\bar{b}\,l^+\nu\,l'^-\,\bar{\nu}'$$

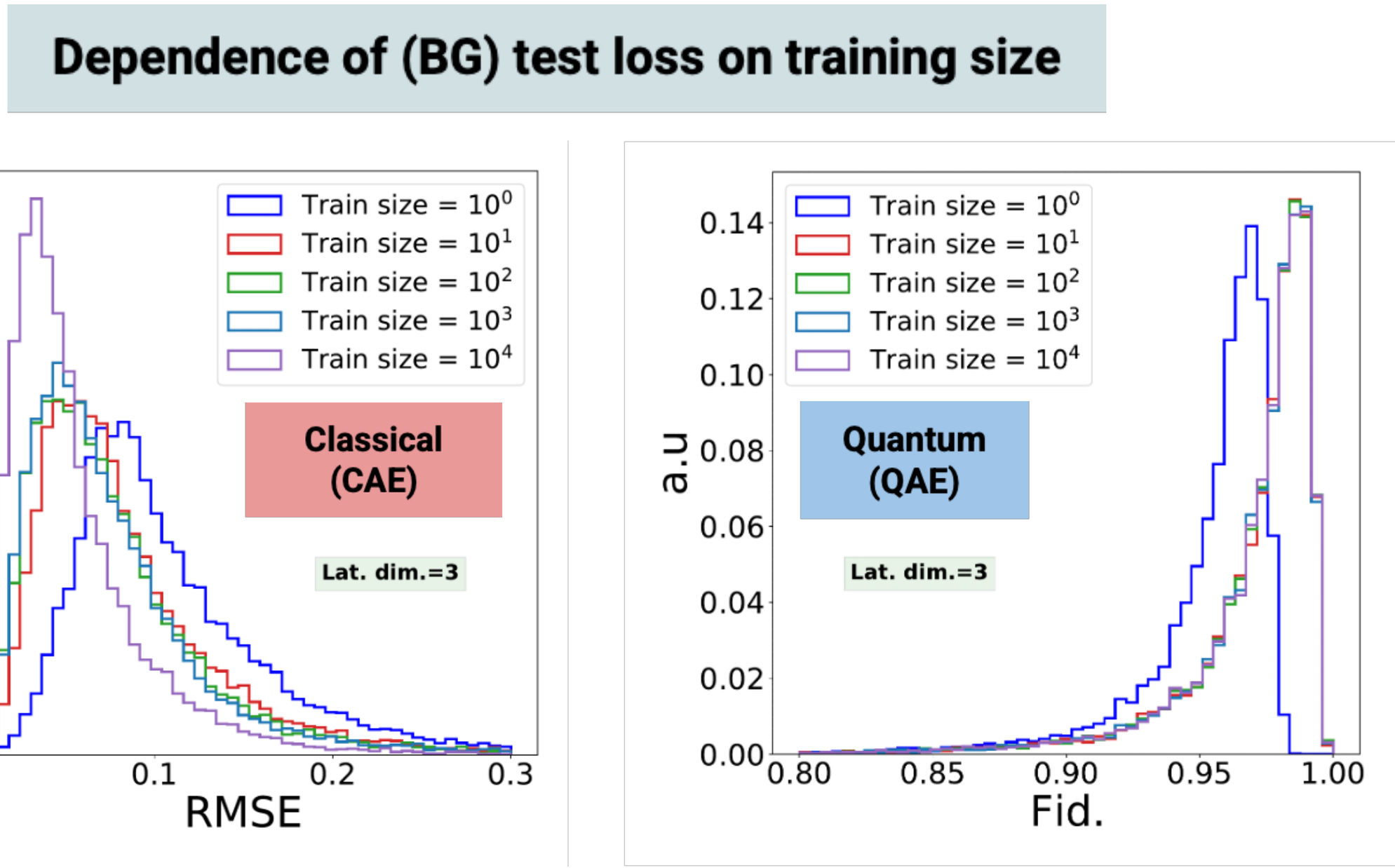
Use four variables: $(p_T^{b_1}, p_T^{l_1}, p_T^{l_2}, MET)$

training only on background -> anomaly detection

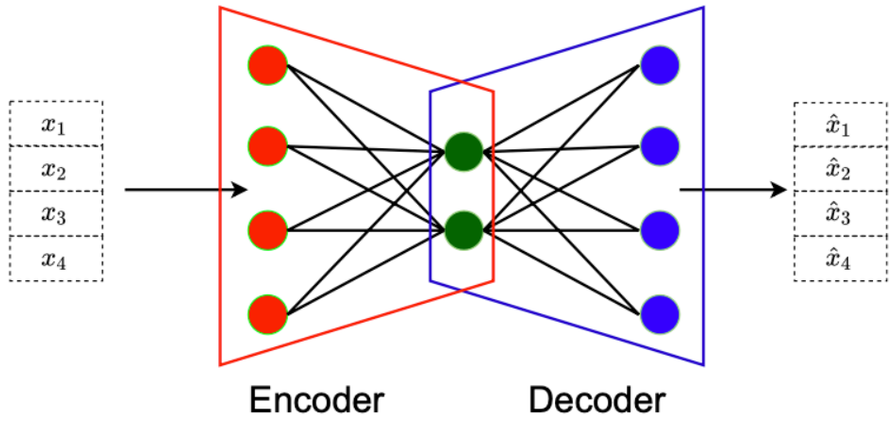
Train a classical autoencoder(CAE) and quantum autoencoder (QAE) with latent spaces: 1,2 and 3 and different sizes of training dataset.

- Also used in (h->inv)jj trained on (Z->inv)jj, with full CAE optimisation

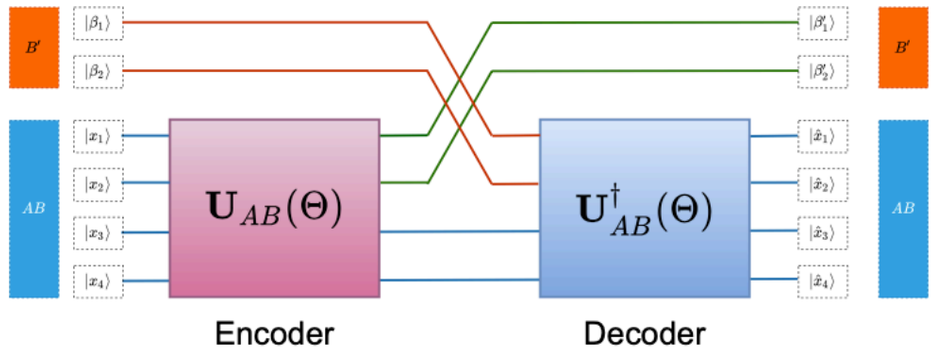
Results: Training size dependence

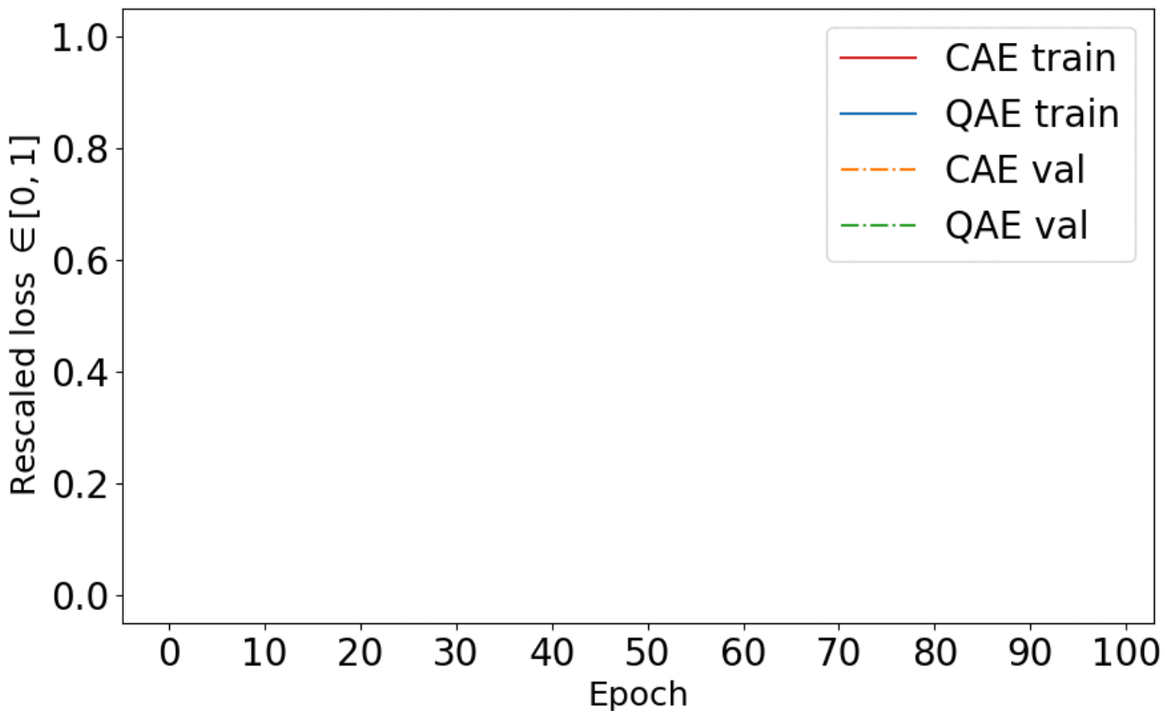


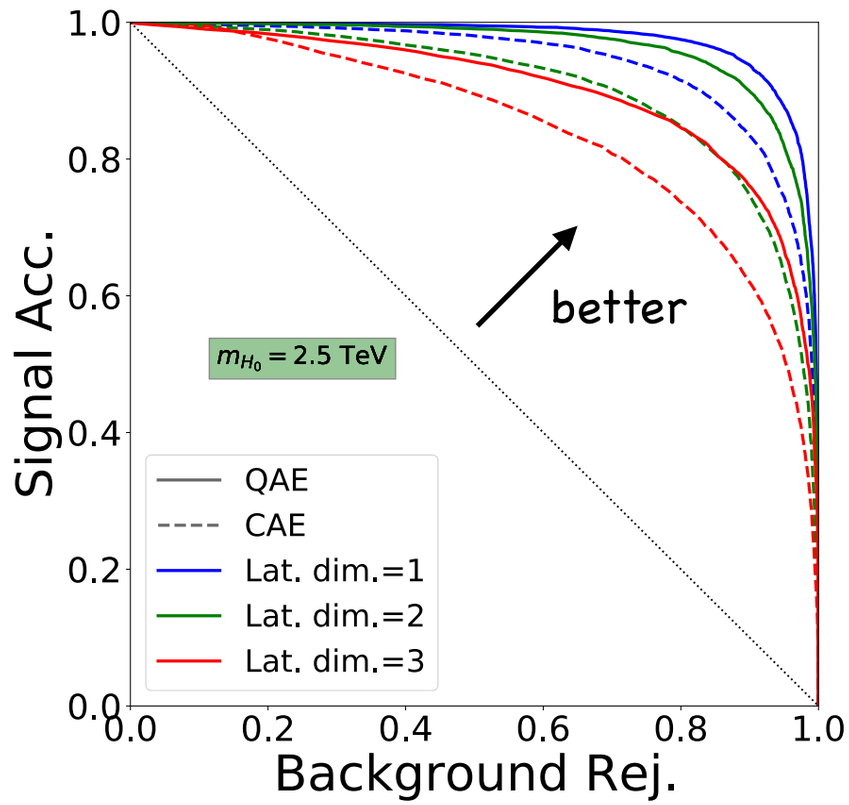
Classical autoencoder



Quantum autoencoder



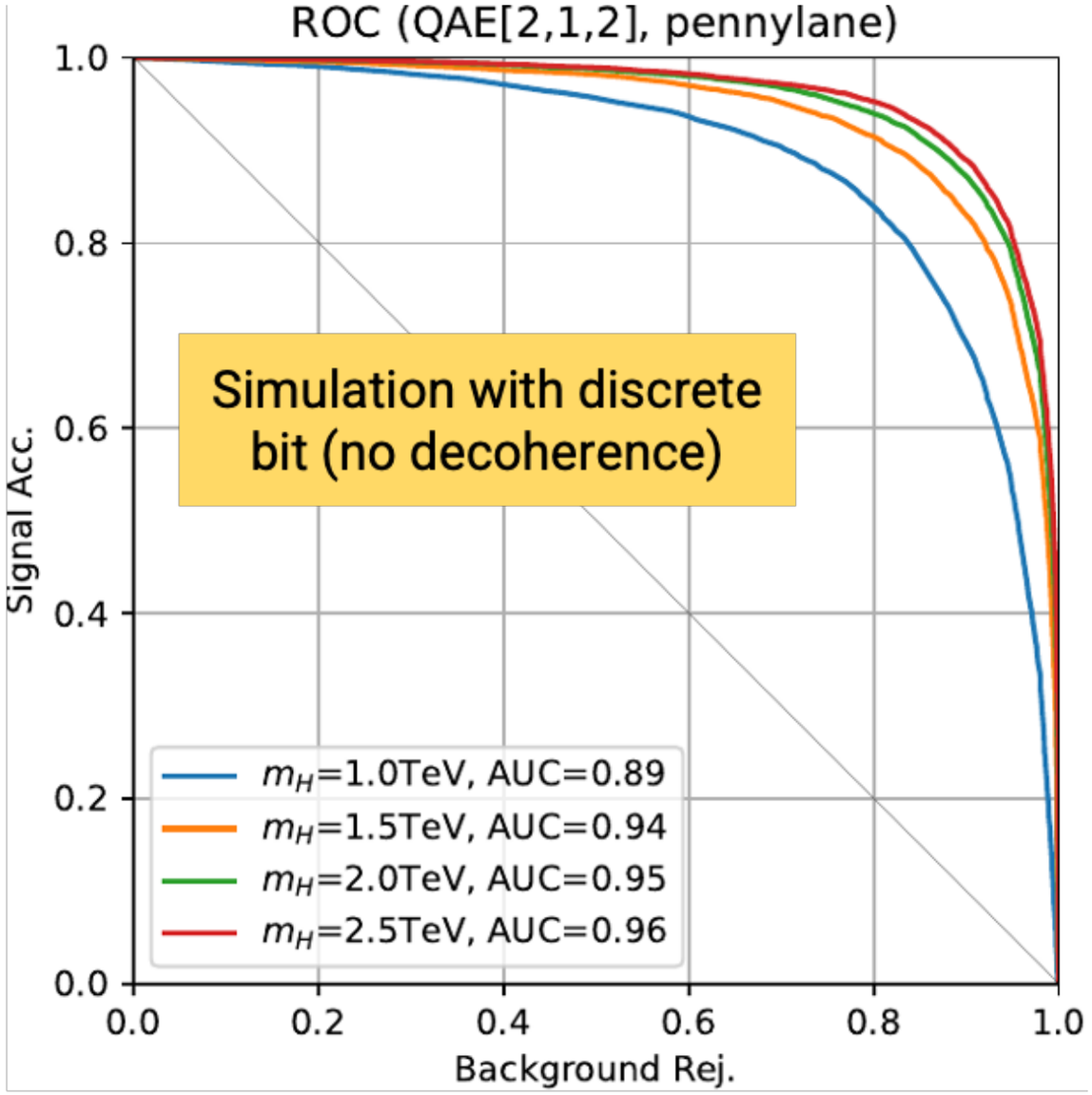


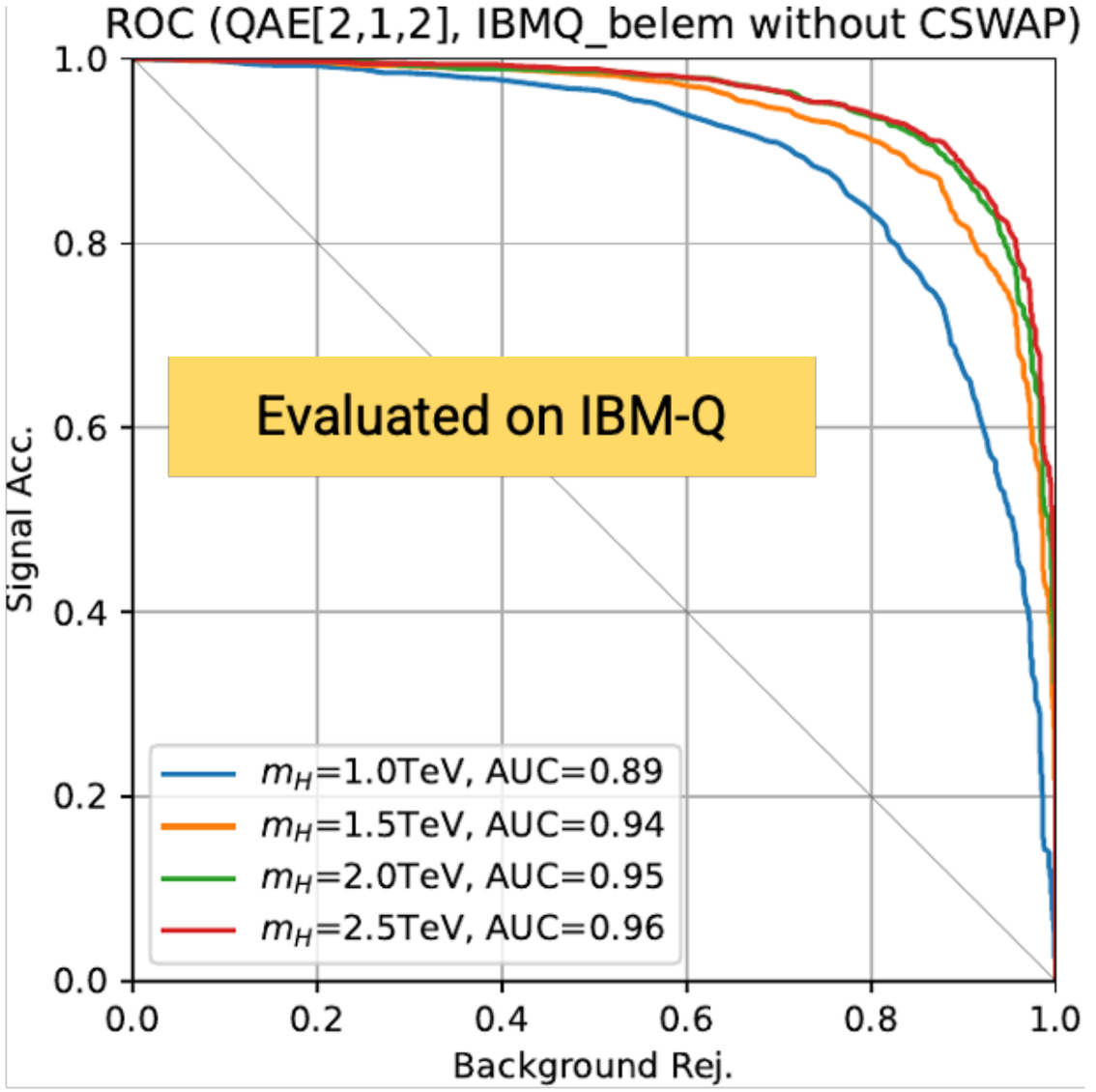


→ Much faster training and better performance for Quantum autoencoder

→ In our test cast, outcome prevails for much larger classical networks

Results: Benchmark on IBM-Q





Generative Algorithms

The probability distribution of a quantum model $p_\theta(x)$ can be interpreted as a quantum expectation value, to wich the shift rule can be applied to calculate the partial derivatives

E.g. consider exp value of function $g(x)$ that takes samples $x \in \mathcal{X}$

$$\mathbb{E}_{x\sim p_\mu}[g(x)] = \int_{\mathcal{X}} p_\mu(x) g(x)\, dx$$

produced by generative model $p_\mu(x)$, that depends on parameters $\mu \in \mathbb{R}$

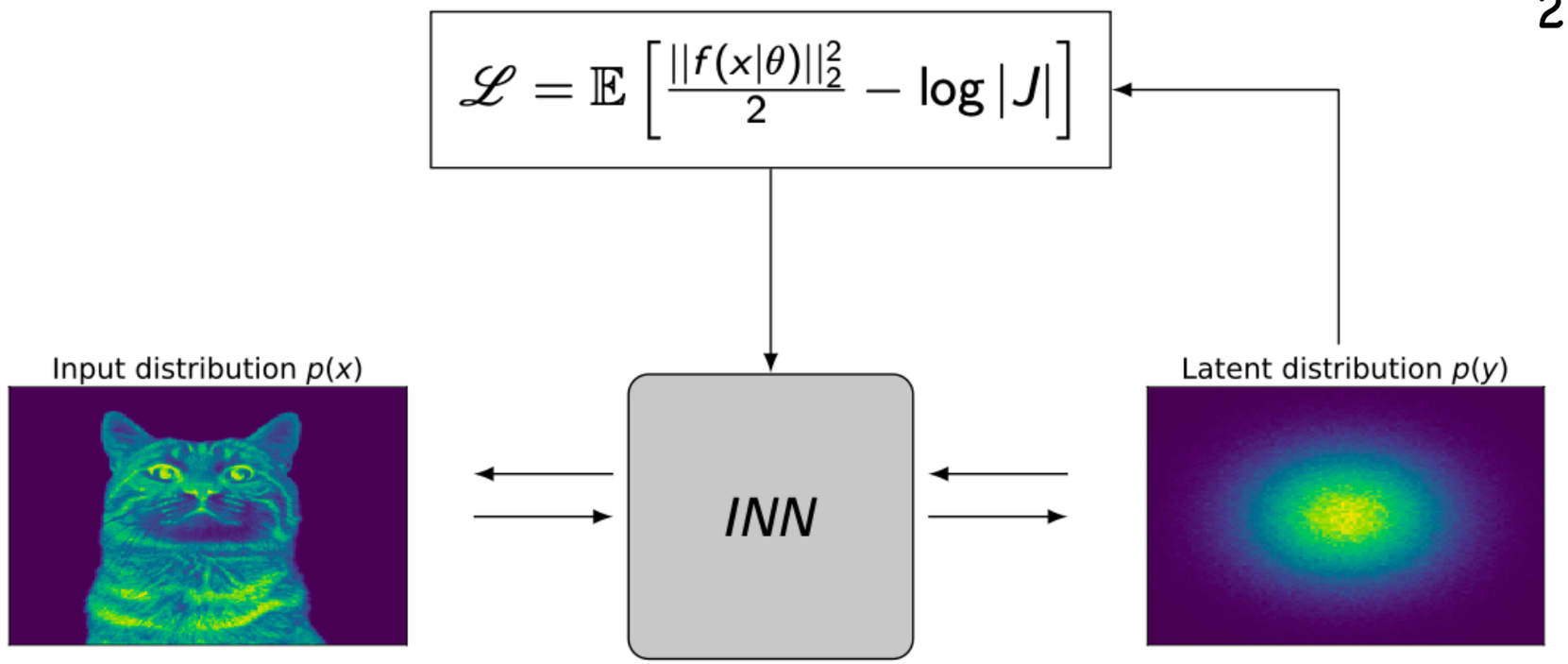
Shift rule gives:

$$\begin{aligned}\partial_\mu \mathbb{E}_{x\sim p_\mu}[g(x)] &= \int_{\mathcal{X}} dx \left(\partial_\mu p_\mu(x)\right) g(x) \\ &= \frac{1}{2\sin(s)} \left(\int_{\mathcal{X}} \left(\partial_\mu p_{\mu-s}(x)\right) g(x)\, dx - \int_{\mathcal{X}} \left(\partial_\mu p_{\mu+s}(x)\right) g(x) \right)\, dx \\ &= \frac{1}{2\sin(s)} \left(\mathbb{E}_{x\sim p_{\mu-s}}[g(x)] - \mathbb{E}_{x\sim p_{\mu+s}}[g(x)]\right)\end{aligned}$$

$\longrightarrow$ Can be evaluated by two-sample testing

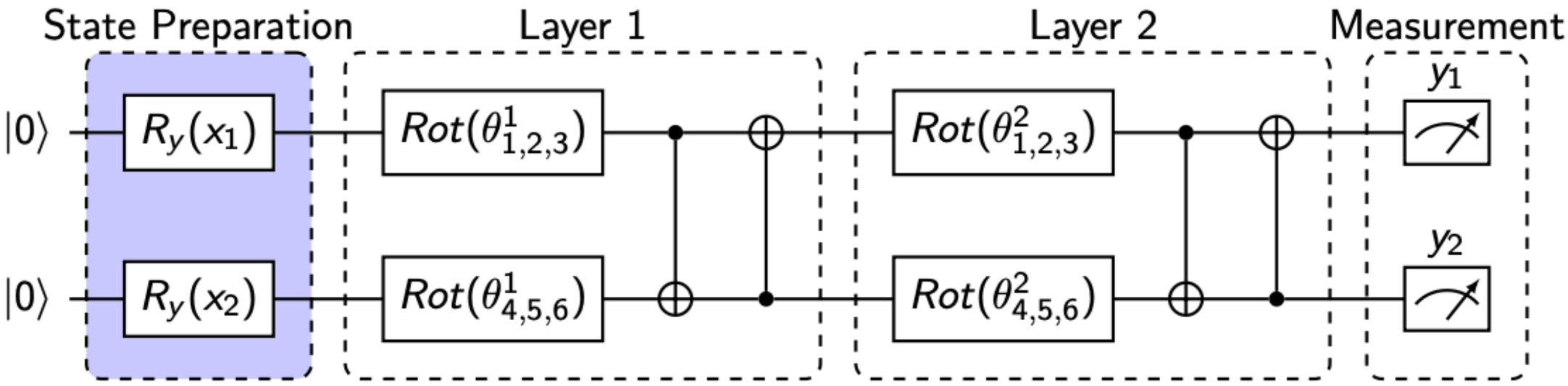
Quantum Invertible Neural Networks
Quantum Normalising Flow

[Rousselot, MS '23
2302.12906]

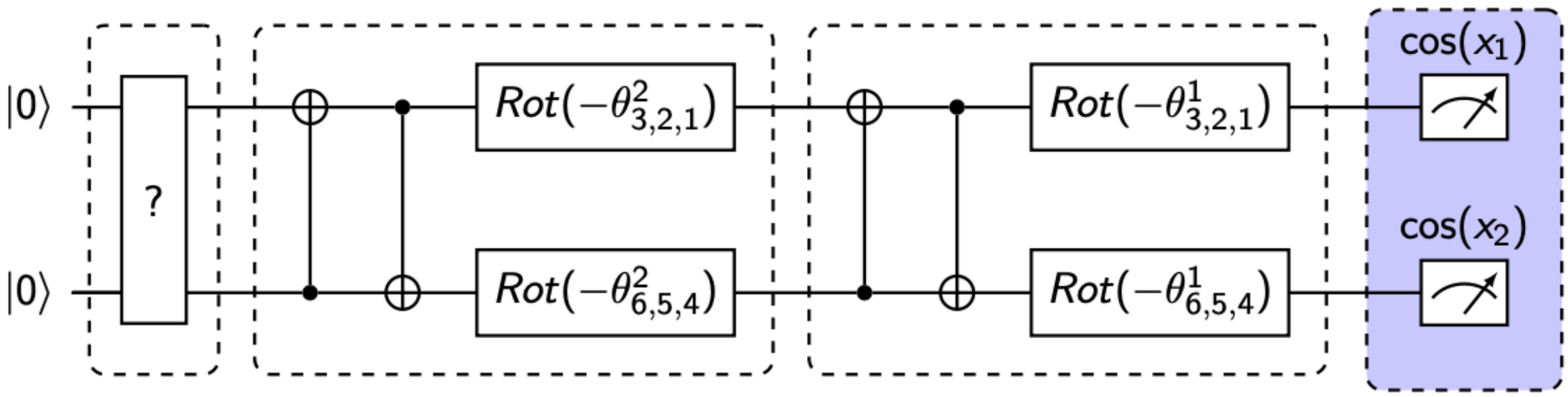


- Train a transformation from input distribution p(x) to gaussian distribution p(y) = N(0,1)
- Can create samples from p(x) by sampling from p(y) and calculating x = Inverse(INN(y))
- Loss function requires jacobian J, which is already available for QNNs via parameter shift rules

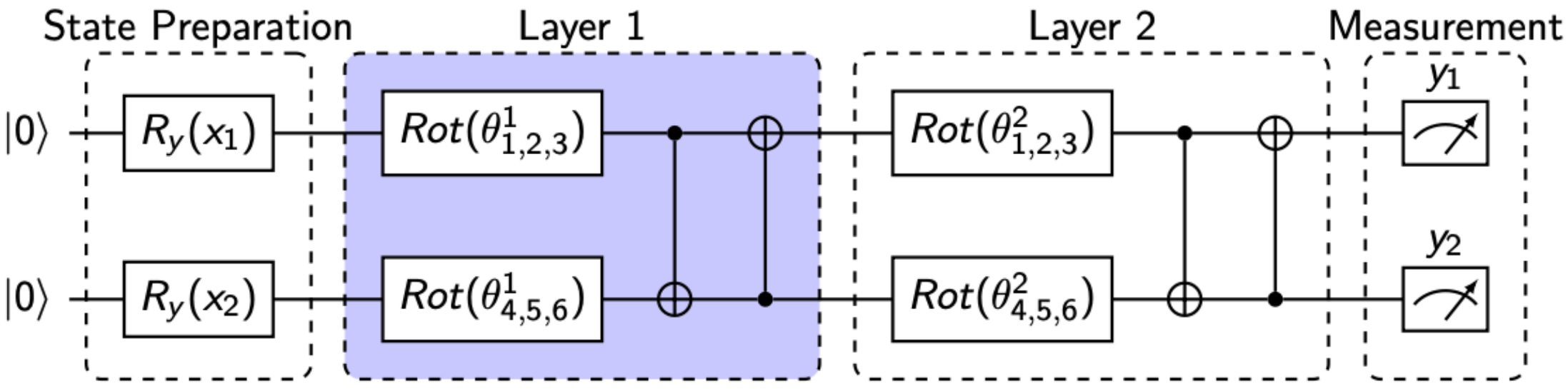
Forward pass:



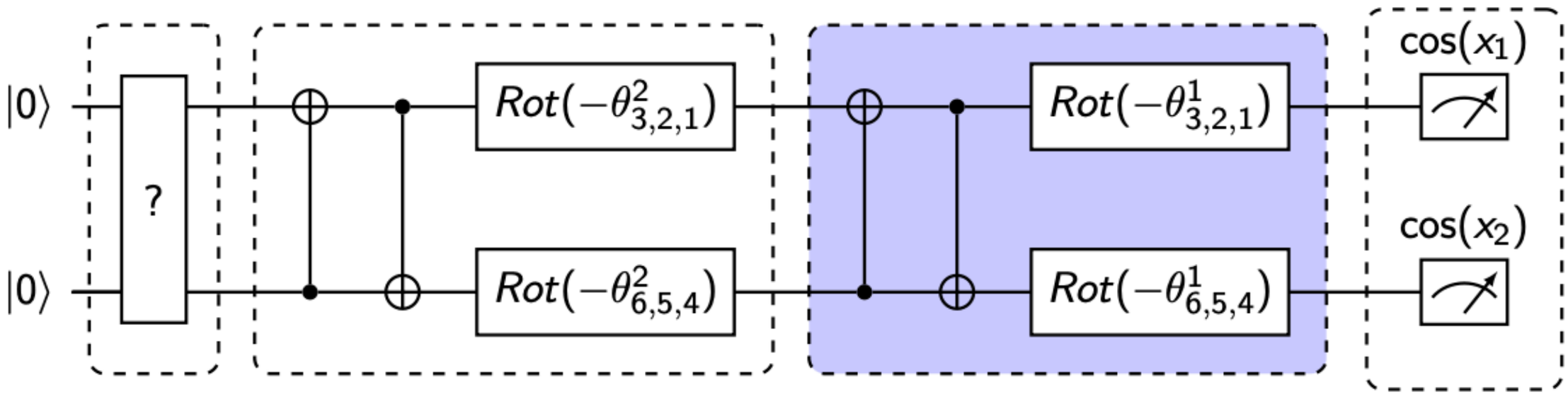
Backward pass:



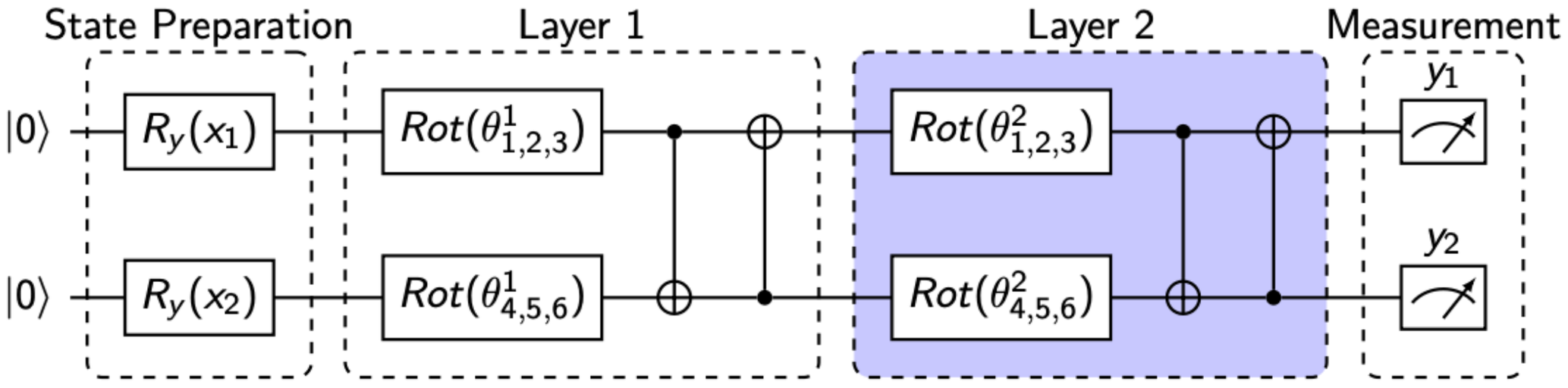
Forward pass:



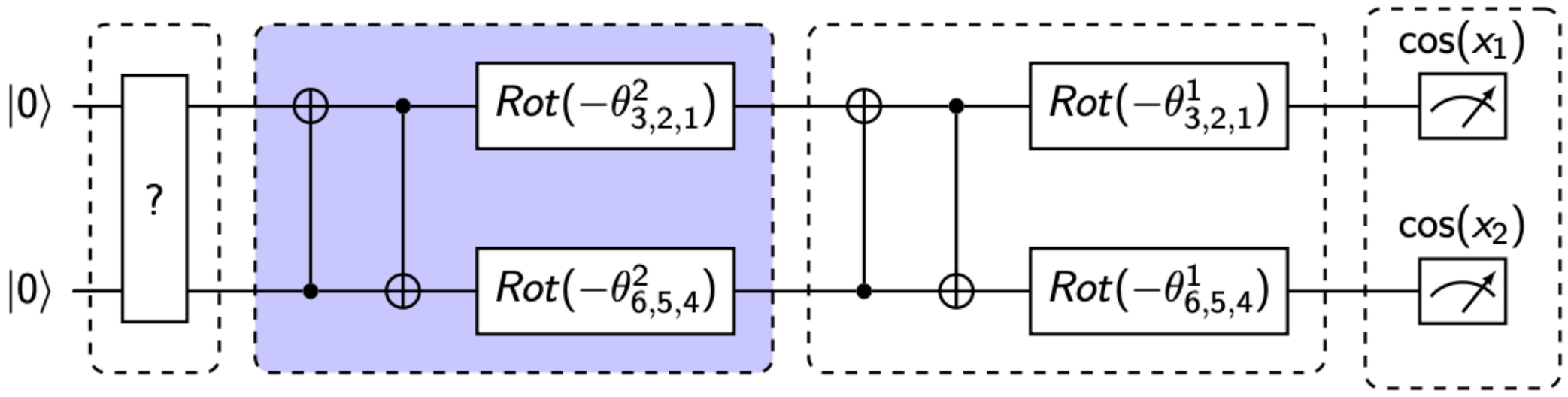
Backward pass:



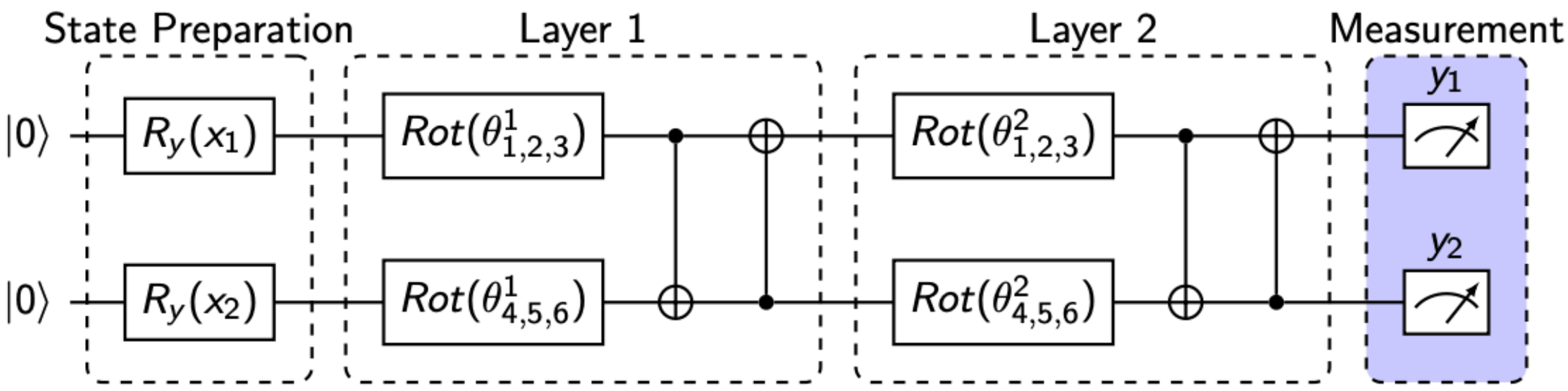
Forward pass:



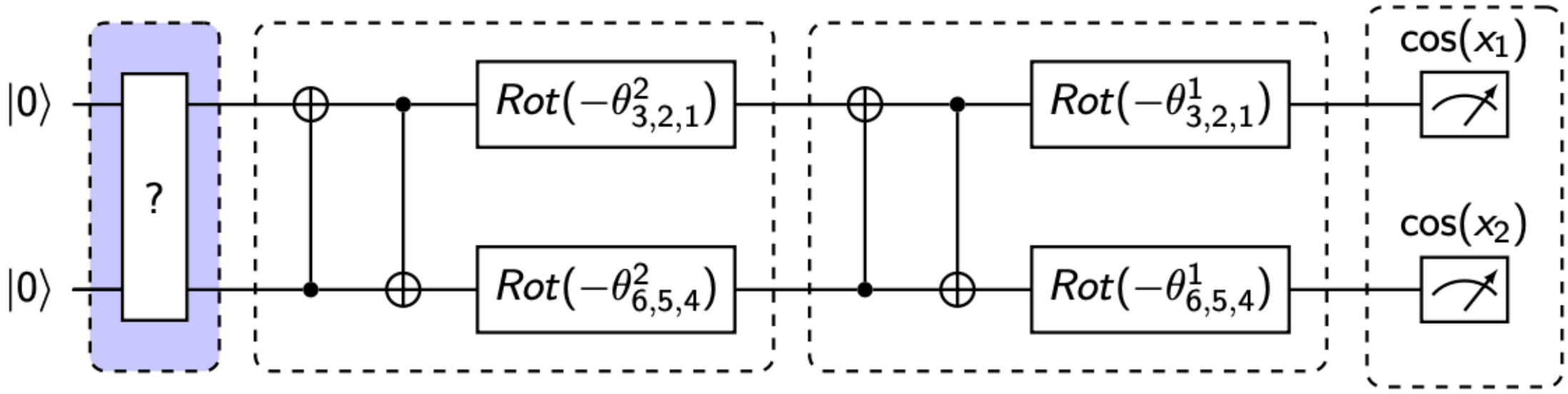
Backward pass:



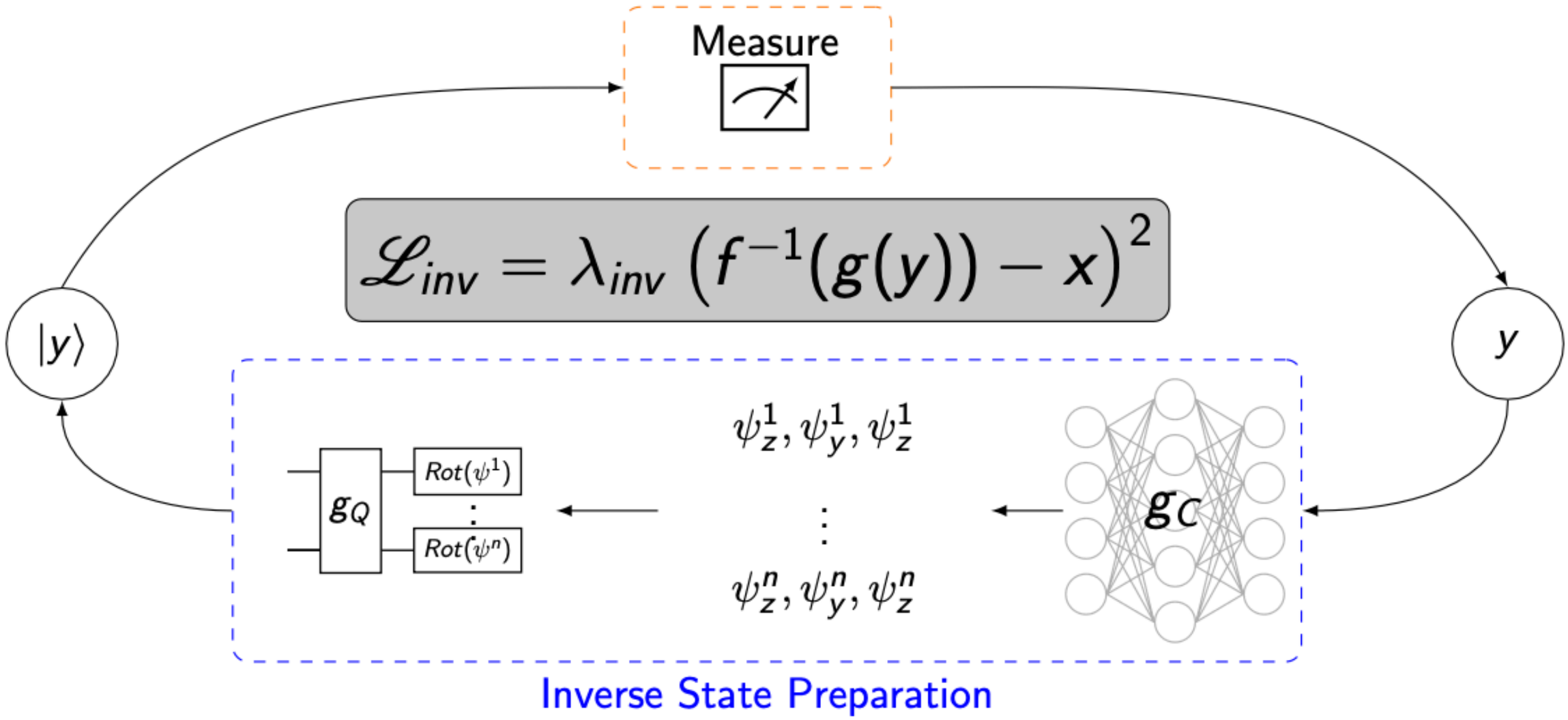
Forward pass:

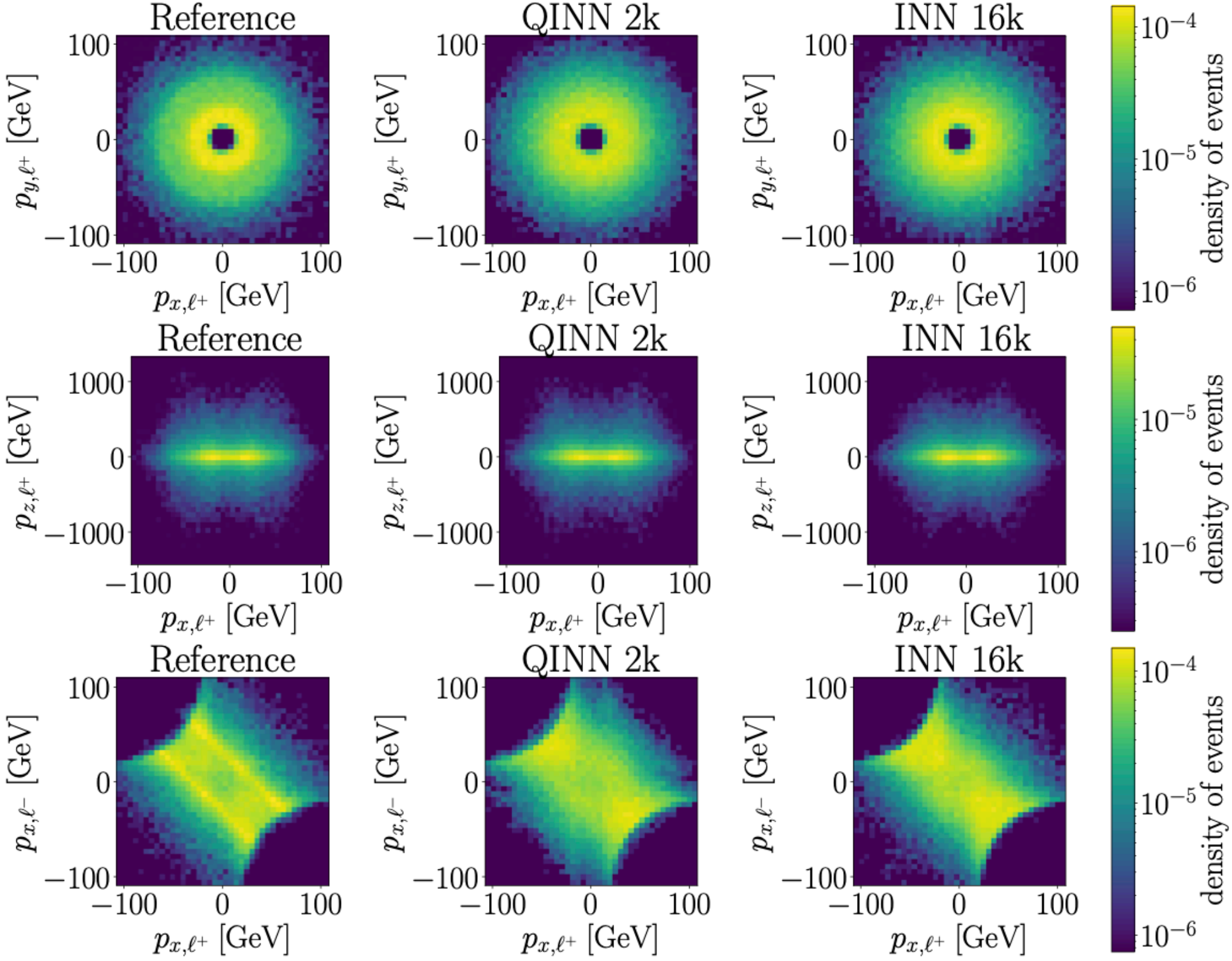


Backward pass:

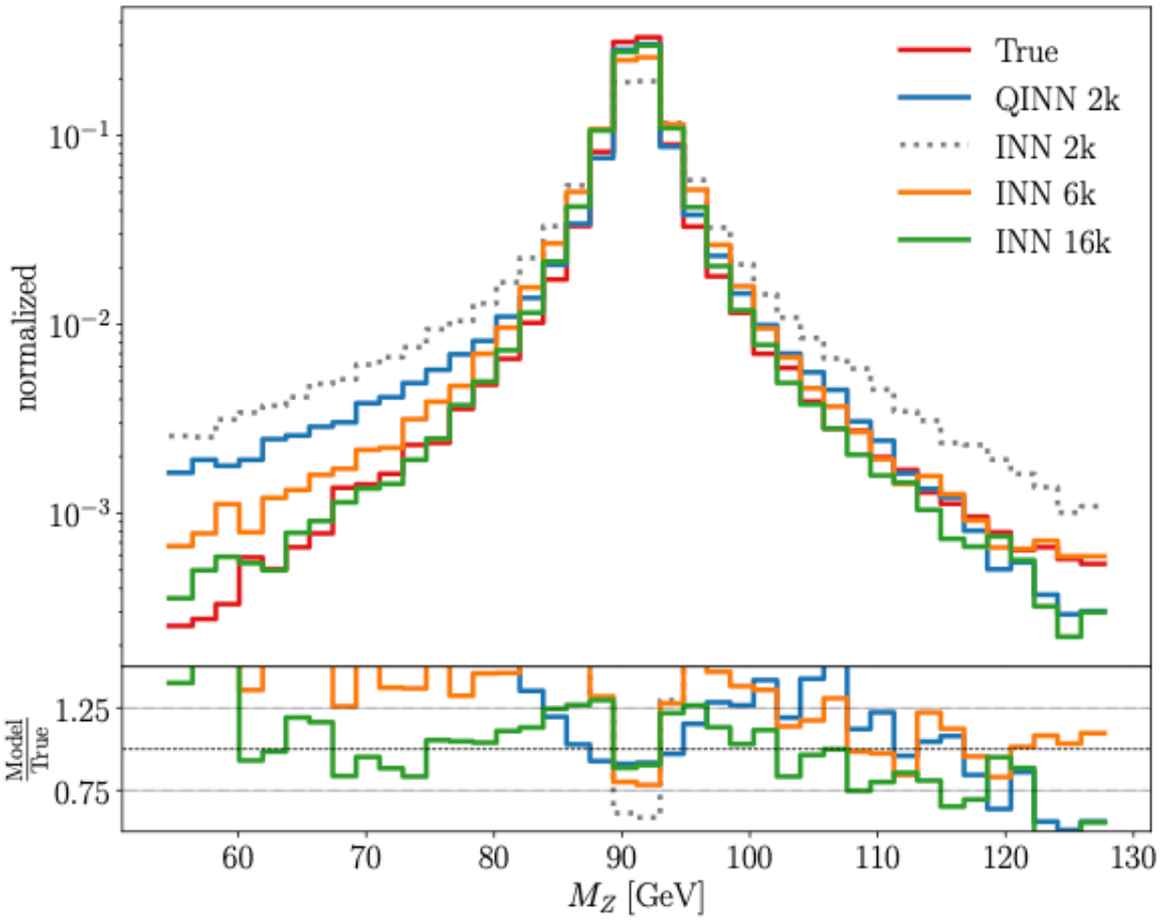


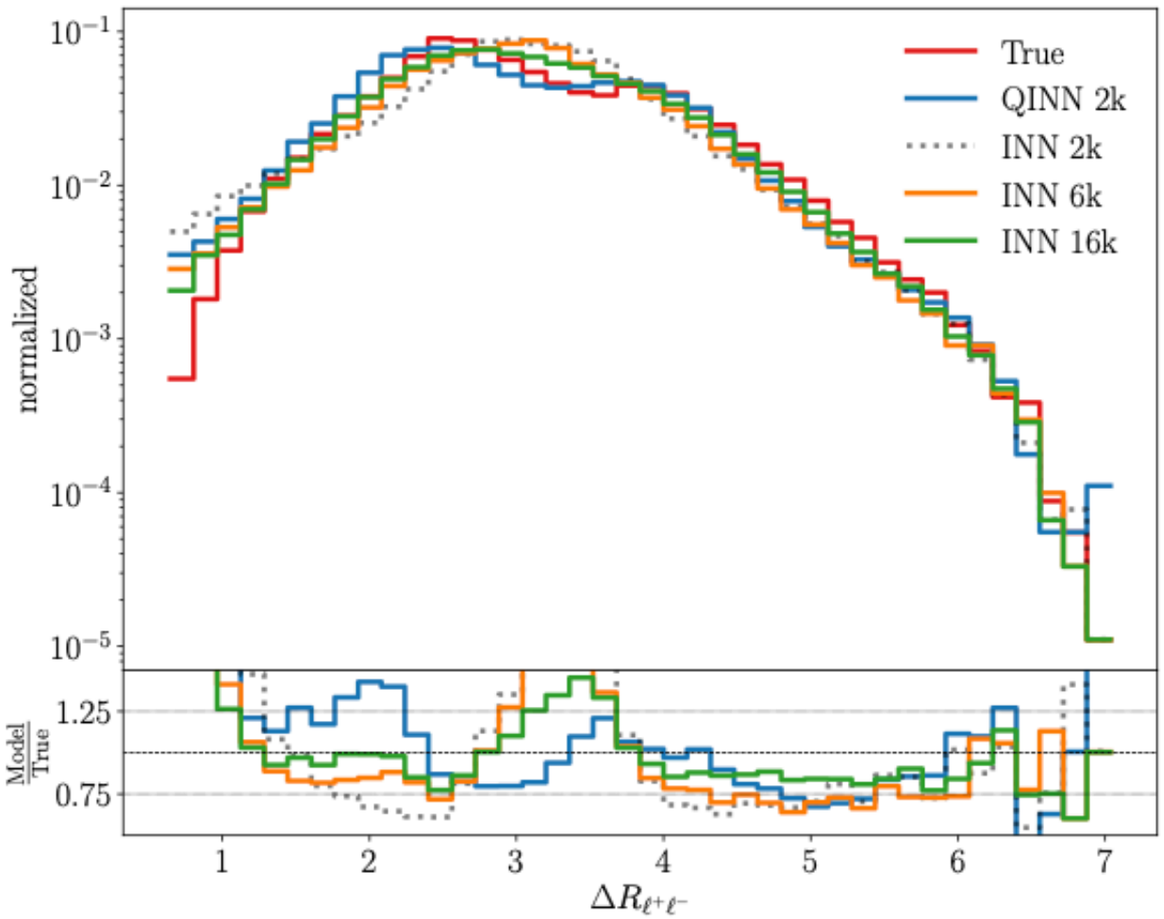
- One solution: Train an **inverse state preparation (ISP)** g : y -> |y> and the model f s.t. $f^{-1}(g(y)) \sim x$




Reference
QINN 2k
INN 16k
p_{y,ℓ^+} [GeV]
p_{x,ℓ^+} [GeV]
p_{z,ℓ^+} [GeV]
p_{x,ℓ^-} [GeV]
density of events

process $pp \to Zj \to \ell^+\ell^- j$





Comparison of QINN with INN of varying size