Introduction to Quantum Machine Learning

Disclaimer: Work in progress. Portions of these written materials are incomplete.

Quantum Computing Basics

Lightning Recap

Origins of Quantum Computing

- Idea: making a computer out of quantum-mechanical building blocks in order to better mimic nature's quantum mechanical properties

- Richard Feynman (1982):
  "Nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical."

Quantum Mechanics Basics

- Quantum theory: a more general form of probability theory
  - Complex amplitudes allow for wave-like interference in the space of possibilities

$$p(x) \in \mathbb{R}^+ \qquad \psi(x) \in \mathbb{C}$$

Qubits as building blocks

Classical bit

Quantum bit (qubit)

$$\psi_0 |0\rangle + \psi_1 |1\rangle$$

$$\begin{pmatrix} \psi_0 \\ \psi_1 \end{pmatrix} \in \mathbb{C}^2$$

$$p(0) + p(1) = 1$$

$$|\psi_0|^2 + |\psi_1|^2 = 1$$

Qubits as building blocks

Single qubit $\cong \mathbb{C}^2$

N qubits $\cong \mathbb{C}^{2^N} \cong \bigotimes_{j=1}^{N} \mathbb{C}^2$

$$\psi_0 |0\rangle + \psi_1 |1\rangle$$

$$\sum_{j\in\{0,1\}^n} \psi_j |j\rangle$$

Superposition of 2^1 values (one bit)

Superposition of 2^N values (bitstrings)

$\mathcal{H} \cong \mathbb{C}^{2^N}$

Understanding quantum circuits

Quantum circuit: sequence of evolutions of a quantum memory state

gates

measurement

Registers

(qubits)

Outcome data

Quantum Artificial Intelligence

Quantum AI: main areas of current interest

Quantum-enhanced optimization:

using quantum dynamics for an optimization advantage

Quantum deep learning:

learning quantum representations for quantum data

Quantum Deep Learning?

Classical vs. Quantum Deep Learning

- Parameters $\mathbf{\Phi}$
- Input $\boldsymbol{x}$
- Feedforward operation f
- Loss function $L(f(\boldsymbol{x}, \mathbf{\Phi}), y)$
- Goal: find $\underset{\mathbf{\Phi}}{\operatorname{argmin}} L(f(\boldsymbol{x}), \mathbf{\Phi}), y)$

- Parameters $\mathbf{\Phi}$
- Input $|\xi_0\rangle$
- Feedforward operation $\hat{U}(\mathbf{\Phi})$
- Loss operator $\hat{L}$
- Goal: find $\underset{\mathbf{\Phi}}{\operatorname{argmin}} \langle\xi_{\mathbf{\Phi}}|\,\hat{L}\,|\xi_{\mathbf{\Phi}}\rangle$

Quantum-Classical Optimization of Quantum Neural Nets

QNN training loop:

1. Inference (sampling) done on QPU
2. Loss expectation estimate relayed to CPU
3. CPU suggests new parameters
4. Repeat 1-3 until convergence

Theory: Quantum Neural Nets & Gradients

1. QNN's general structure
2. Pauli operator expansion*
3. Quantum exponential compilation*
4. Quantum expectation estimation
5. Finite-difference gradients
6. Parameter-shift gradients*
7. Stochastic Gradient estimation*

*see TFQ whitepaper for detailed background

Theory: QNN's & Gradients

- Multi-layer QNN's: compilation hierarchy

$$\hat{U}(\boldsymbol{\theta}) = \prod_{\ell=1}^{L} \hat{V}^{\ell} \hat{U}^{\ell}(\boldsymbol{\theta}^{\ell})$$

$$\hat{U}^{\ell}(\boldsymbol{\theta}^{\ell}) \equiv \bigotimes_{j=1}^{M_{\ell}} \hat{U}_j^{\ell}(\theta_j^{\ell})$$

$$\hat{U}_j^{\ell}(\theta_j^{\ell}) = e^{-i\theta_j^{\ell}\hat{g}_j^{\ell}}, \qquad \hat{g}_j^{\ell} = \sum_{k=1}^{K_{j\ell}} \beta_k^{j\ell} \hat{P}_k,$$

$$\hat{U}_j^{\ell}(\theta_j^{\ell}) = \prod_k e^{-i\theta_j^{\ell}\beta_k^{j\ell}\hat{P}_k}$$

$$\hat{U}_j^{\ell}(\theta_j^{\ell}) = \prod_k \left[\cos(\theta_j^{\ell}\beta_k^{j\ell})\hat{I} - i\sin(\theta_j^{\ell}\beta_k^{j\ell})\hat{P}_k\right]$$

Theory: QNN's & Gradients

- Quantum Expectation estimation

$$f(\boldsymbol{\theta}) = \langle \hat{H} \rangle_{\boldsymbol{\theta}} \equiv \langle \Psi_0 | \hat{U}^{\dagger}(\boldsymbol{\theta}) \hat{H} \hat{U}(\boldsymbol{\theta}) | \Psi_0 \rangle$$

$$\hat{H} = \sum_{k=1}^{N} \alpha_k \hat{h}_k \equiv \boldsymbol{\alpha} \cdot \hat{\boldsymbol{h}}, \qquad \hat{h}_j = \sum_{k=1}^{J_j} \gamma_k^j \hat{P}_k,$$

$$f(\boldsymbol{\theta}) = \langle \hat{H} \rangle_{\boldsymbol{\theta}} = \sum_{k=1}^{N} \alpha_k \langle \hat{h}_k \rangle_{\boldsymbol{\theta}} \equiv \boldsymbol{\alpha} \cdot \boldsymbol{h}_{\boldsymbol{\theta}},$$

- Can estimate expectation values of observables by sampling the output of the quantum computer in various bases

Theory: QNN's & Gradients

- Finite-difference gradients

$$f(\boldsymbol{\theta}) = \langle \hat{H} \rangle_{\boldsymbol{\theta}} \equiv \langle \Psi_0 | \, \hat{U}^\dagger(\boldsymbol{\theta}) \hat{H} \hat{U}(\boldsymbol{\theta}) \, | \Psi_0 \rangle$$

$$\partial_k f(\boldsymbol{\theta}) = \frac{f(\boldsymbol{\theta} + \varepsilon \boldsymbol{\Delta}_k) - f(\boldsymbol{\theta} - \varepsilon \boldsymbol{\Delta}_k)}{2\varepsilon} + \mathcal{O}(\varepsilon^2)$$

- Better ways to find gradients?
  - Use chain rule + the fact that QNN's are expressible as composition of exponentials
  - see TFQ whitepaper

Theory: QNN's & Gradients

see TFQ whitepaper

- Parameter shift rule

$$\hat{U}^{\ell}(\boldsymbol{\theta}^{\ell}) \equiv \bigotimes_{j=1}^{M_\ell} \hat{U}_j^{\ell}(\theta_j^{\ell})$$

$$\hat{U}_j^{\ell}(\theta_j^{\ell}) = \prod_k e^{-i\theta_j^{\ell}\beta_k^{j\ell}\hat{P}_k}$$

$$\hat{U}^{\ell}(\boldsymbol{\theta}^{\ell}) \mapsto \hat{U}^{\ell}(\boldsymbol{\eta}^{\ell}) \equiv \bigotimes_{j\in\mathcal{I}_\ell}\left(\prod_k e^{-i\eta_k^{j\ell}\hat{P}_k}\right)$$

$$\eta_k^{j\ell} \equiv \theta_j^{\ell}\beta_k^{j\ell}$$

$$f(\boldsymbol{\eta}) \equiv \langle\Psi_0|\,\hat{U}^{\dagger}(\boldsymbol{\eta})\hat{H}\hat{U}(\boldsymbol{\eta})\,|\Psi_0\rangle$$

$$\frac{\partial f}{\partial\theta_j^{\ell}} = \sum_k \frac{\partial f}{\partial\eta_k^{j\ell}}\frac{\partial\eta_k^{j\ell}}{\partial\theta_j^{\ell}} = \sum_k \beta_k^{j\ell}\frac{\partial f}{\partial\eta_k}$$

$$e^{-i\eta_k^{j\ell}\hat{P}_k} = \cos(\eta_k^{j\ell})\hat{I} - i\sin(\eta_k^{j\ell})\hat{P}_k$$

$$\frac{\partial}{\partial\eta_k^{j\ell}}f(\boldsymbol{\eta}) = f(\boldsymbol{\eta} + \tfrac{\pi}{4}\boldsymbol{\Delta}_k^{j\ell}) - f(\boldsymbol{\eta} - \tfrac{\pi}{4}\boldsymbol{\Delta}_k^{j\ell})$$

Theory: QNN's & Gradients

see TFQ whitepaper

- Stochastic gradient estimation

$$\frac{\partial f}{\partial \theta_j^\ell} = \sum_{k=1}^{K_{j\ell}} \beta_k^{j\ell} \frac{\partial f}{\partial \eta_k} = \sum_{k=1}^{K_{j\ell}} \Big[\sum_{\pm} \pm \beta_k^{j\ell} f(\boldsymbol{\eta} \pm \tfrac{\pi}{4} \boldsymbol{\Delta}_k^{j\ell}) \Big] \qquad k \sim \Pr(k|j,\ell) = |\beta_k^{j\ell}| / (\textstyle\sum_{o=1}^{K_{j\ell}} |\beta_o^{j\ell}|)$$

- Doubly stochastic gradient estimation

$$f(\boldsymbol{\theta}) = \langle \hat{H} \rangle_{\boldsymbol{\theta}} = \sum_{m=1}^{N} \alpha_m \langle \hat{h}_m \rangle_{\boldsymbol{\theta}} = \sum_{m=1}^{N} \sum_{q=1}^{J_m} \alpha_m \gamma_q^m \langle \hat{P}_{qm} \rangle_{\boldsymbol{\theta}},$$

$$\{q, m\} \sim \Pr(q, m) = |\alpha_m \gamma_q^m| / (\textstyle\sum_{d=1}^{N} \sum_{r=1}^{J_d} |\alpha_d \gamma_r^d|)$$

- Triply stochastic gradient descent

$$\theta_j^\ell \sim \Pr(j, \ell) = \textstyle\sum_{k=1}^{K_{j\ell}} |\beta_k^{j\ell}| / (\sum_{u=1}^{L} \sum_{i=1}^{M_u} \sum_{o=1}^{K_{iu}} |\beta_o^{iu}|)$$

$$\{j, \ell, k, q, m\} \sim \Pr(j, \ell, k, q, m) = \Pr(k|j, \ell) \Pr(j, \ell) \Pr(q, m)$$

Example: a simple quantum state classifier

- App: classifying input quantum states

- Data:

Notebook link

- Model:

Example: a naive MNIST quantum classifier

- App: classifying input classical data encoded in quantum states

- Data:
- Model:

[Notebook link]

How to more practically leverage quantum computing power?

Hybridize Classical & Quantum Deep Learning!

Hybrid Quantum-classical neural networks

Combine quantum and classical representation power

Hybrid Quantum-classical neural networks

- Practical approach to add quantum components to classical deep learning representations
- Can make hybrid computational graphs from new building blocks

Hybrid quantum-classical computational graph

QNN: quantum neural network

DNN: (Classical) Deep Neural Network

Quantum-classical Hybrid neural networks & hybrid backprop

- Feedforward expectation value vector: $(\boldsymbol{h_\theta})_k = \langle \hat{h}_k \rangle_{\boldsymbol{\theta}} \equiv \langle \Psi_0 | \hat{U}^\dagger(\boldsymbol{\theta}) \hat{h}_k \hat{U}(\boldsymbol{\theta}) | \Psi_0 \rangle$

- Effective backpropagated error Hamiltonian:
  - Gradients can be estimated with gradient methods from last lecture

$$\hat{H}_{\boldsymbol{g}} \equiv \sum_k g_k \hat{h}_k$$

$$\frac{\partial}{\partial \theta_j} \langle \hat{H}_{\boldsymbol{g}} \rangle_{\boldsymbol{\theta}} = \frac{\partial}{\partial \theta_j} (\boldsymbol{g} \cdot \boldsymbol{h_\theta}) = \sum_k g_k \frac{\partial h_{\boldsymbol{\theta},k}}{\partial \theta_j}$$

Reminder : Types of Machine Learning

ML: Algorithms to identify patterns in data

Generative models

Dataset

$$\mathcal{D} = \{\boldsymbol{x}_k\}_{k=1}^{|\mathcal{D}|} \quad \boldsymbol{x}_k \sim p_{\text{true}}(\boldsymbol{x})$$

Want to learn:

$$q_{\Phi}(\boldsymbol{x}) \approx p_{\text{true}}(\boldsymbol{x})$$

Discriminative models

Dataset

$$\mathcal{D} = \{\boldsymbol{x}_k, \boldsymbol{y}_k\}_k$$

Want to learn:

$$q_{\boldsymbol{\Phi}}(\boldsymbol{y}|\boldsymbol{x}) \approx p_{\text{true}}(\boldsymbol{y}|\boldsymbol{x})$$

$$q_{\boldsymbol{\Phi}}(\boldsymbol{y}|\boldsymbol{x}) = \delta(y - f_{\boldsymbol{\Phi}}(\boldsymbol{x}))$$

Types of Hybrid Quantum-Classical Deep Learning Models

Generative:

e.g. quantum-probabilistic generative models

Discriminative:

e.g. quantum-classical feedforward neural network classifiers

Hybrid Quantum-Classical (HQC)

Models + Software

HQC neural network models:

Breaking up the task of quantum representation learning into quantum & classical components

TensorFlow Quantum:

Integrating quantum capabilities into SOTA ML workflows, with a focus on scalability & performance

Hybrid Quantum-Classical (HQC)

Models + Software

HQC neural network models:

Breaking up the task of quantum representation learning into quantum & classical components

TensorFlow Quantum:

Integrating quantum capabilities into SOTA ML workflows, with a focus on scalability & performance

tensorflow.org/quantum

TensorFlow Quantum

an open source library for the rapid prototyping of hybrid quantum-classical models for classical or quantum data.

Some key features:

- Deep hybridization of TF and Cirq
- Automated expectation & gradient calculations
- Parallelizable high-performance simulation with TFQ-qSim
- Accelerating any quantum variational algo workflow

tensorflow.org/quantum

TensorFlow Quantum

an open source library for the rapid prototyping of hybrid quantum-classical models for classical or quantum data.

Some key features:

- Focused on aiding the construction & training of *hybrid quantum-classical models*
- Focused on supporting quantum data *(standard quantum datasets coming soon)*

tensorflow.org/quantum

TensorFlow Quantum

an open source library for the rapid prototyping of hybrid quantum-classical models for classical or quantum data.

Some key features:

- Hybrid quantum-classical backprop
- Choices of differentiators
  - Finite-difference
  - Parameter shift
  - Stochastic parameter shift
  - Adjoint diff *(new)*

$$(\boldsymbol{h}_{\boldsymbol{\theta}})_k = \langle \hat{h}_k \rangle_{\boldsymbol{\theta}} \equiv \langle \Psi_0 | \hat{U}^\dagger(\boldsymbol{\theta}) \hat{h}_k \hat{U}(\boldsymbol{\theta}) | \Psi_0 \rangle$$

$$\hat{H}_{\boldsymbol{g}} \equiv \sum_k g_k \hat{h}_k \qquad \frac{\partial}{\partial \theta_j} \langle \hat{H}_{\boldsymbol{g}} \rangle_{\boldsymbol{\theta}} = \frac{\partial}{\partial \theta_j} (\boldsymbol{g} \cdot \boldsymbol{h}_{\boldsymbol{\theta}}) = \sum_k g_k \frac{\partial h_{\boldsymbol{\theta},k}}{\partial \theta_j}$$

tensorflow.org/quantum

TensorFlow Quantum

an open source library for the rapid prototyping of hybrid quantum-classical models for classical or quantum data.

Some key features:

- Deep integration into TF
  - allows for high-performance interfacing with other TF-based frameworks (e.g. TF Probability & TensorBoard)

References

Be sure to check out

tensorflow.org/quantum

[arXiv:2003.02989]

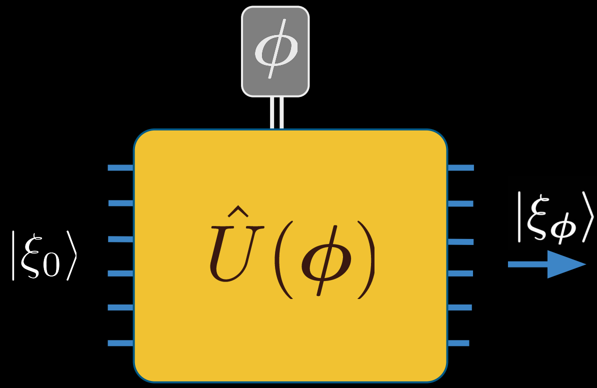
[arXiv:1910.02071]

Novel hybrid approach: Quantum-probabilistic Hybrid DL

Pure Quantum Neural Network Model

Previous solutions

Previous quantum neural nets process strictly pure states



Initial state

$$|\xi_0\rangle$$

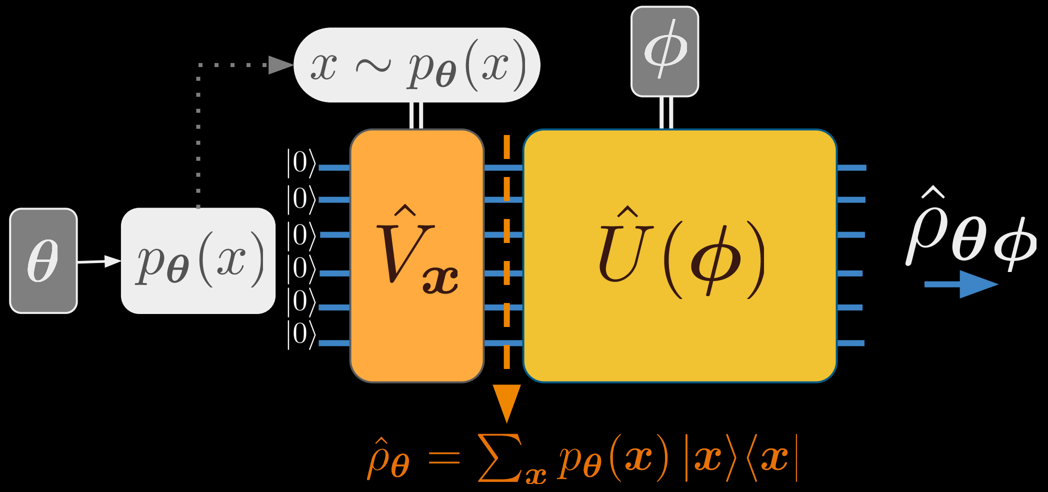
Output state

$$|\xi_\phi\rangle = \hat{U}(\phi)\,|\xi_0\rangle$$

Quantum-probabilistic Hybrid Models

Novel solution:

Combining classical probabilistic inference with quantum neural nets



Latent representation

$$\hat{\rho}_{\boldsymbol{\theta}} = \sum_{x \in \Omega} p_{\boldsymbol{\theta}}(\boldsymbol{x}) |\boldsymbol{x}\rangle\langle\boldsymbol{x}|$$

Visible representation

$$\hat{\rho}_{\boldsymbol{\theta}\boldsymbol{\phi}} = \hat{U}(\boldsymbol{\phi}) \hat{\rho}_{\boldsymbol{\theta}} \hat{U}^{\dagger}(\boldsymbol{\phi})$$

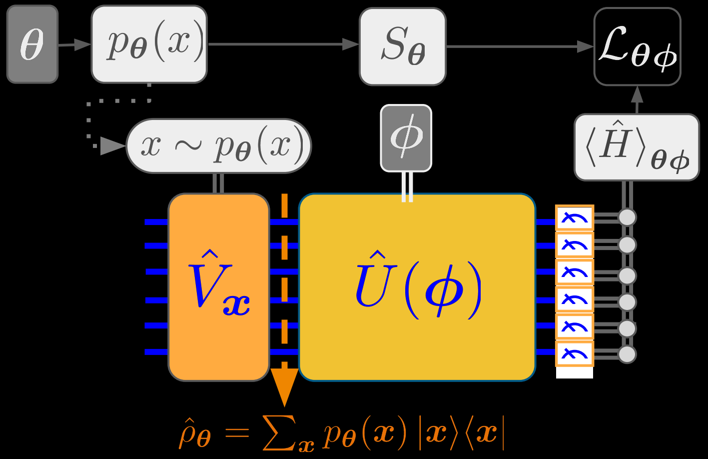
So what can one do with these?

Applications of Hybrid Quantum-probabilistic models

Variational Quantum Thermalization:

Variationally learning how to create the thermal state given a Hamiltonian

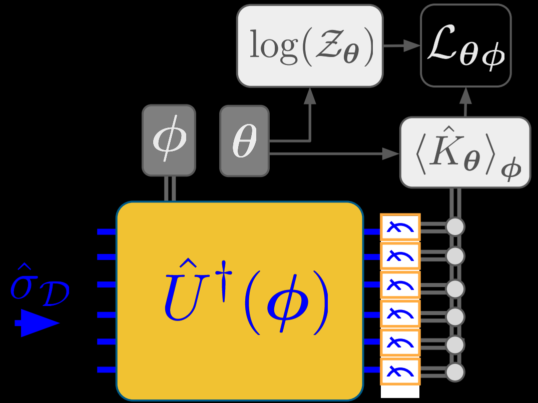
- minimize quantum free energy
- ∃ parameter-shift gradients for θ & ϕ



Quantum Modular Hamiltonian learning:

Learning to replicate mixed data as a parameterized thermal state

- minimize quantum cross-entropy
- ∃ parameter-shift gradients for θ & ϕ



Variational Quantum Thermalization

Results

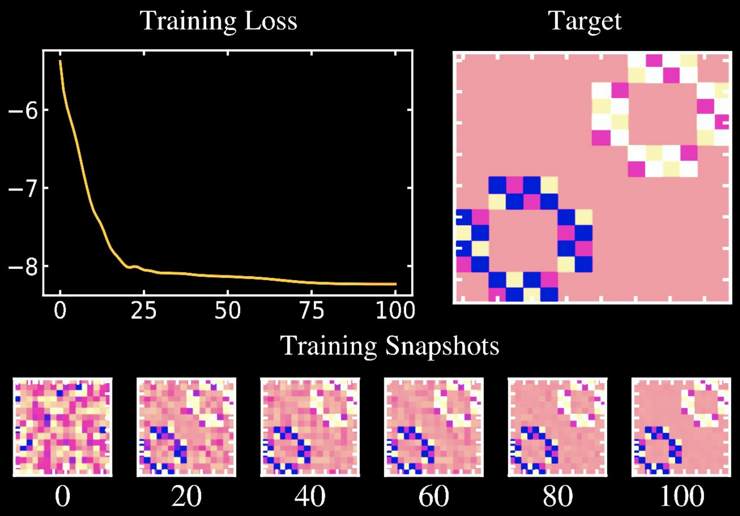
Simulating toy models of superconductors

d-wave superconductor

fermionic Hamiltonian

$$\hat{H}_{d_{x^2-y^2}} = -t \sum_{\langle i,j\rangle,\sigma} (\hat{a}^{\dagger}_{i,\sigma}\hat{a}_{j,\sigma} + \hat{a}^{\dagger}_{j,\sigma}\hat{a}_{i,\sigma})$$

$$\Delta \sum_{\langle i,j\rangle} (\hat{a}^{\dagger}_{i,\uparrow}\hat{a}^{\dagger}_{j,\downarrow} - \hat{a}^{\dagger}_{i,\downarrow}\hat{a}^{\dagger}_{j,\uparrow} + \mathrm{h.c.})$$



Learning to generate a thermal state (covariance matrix plot)

Variational Quantum Thermalization

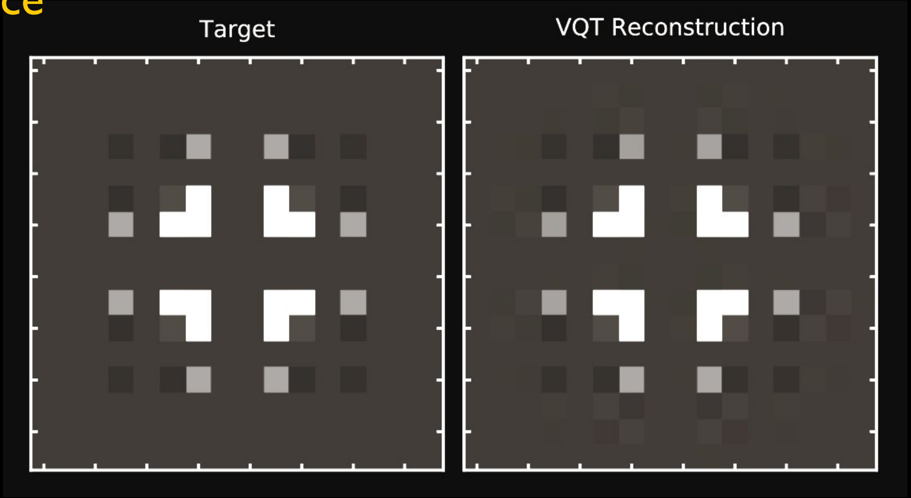
Results

Generating thermal states of 2D spin lattice

2D Heisenberg model

$$\hat{H}_{\mathrm{HEIS}} = \sum_{\langle ij\rangle_h} J_h \hat{\boldsymbol{S}}_i \cdot \hat{\boldsymbol{S}}_j + \sum_{\langle ij\rangle_v} J_v \hat{\boldsymbol{S}}_i \cdot \hat{\boldsymbol{S}}_j$$

Latent model: product of Bernoullis



Generated thermal state

(density matrix)

[after 200 training steps.

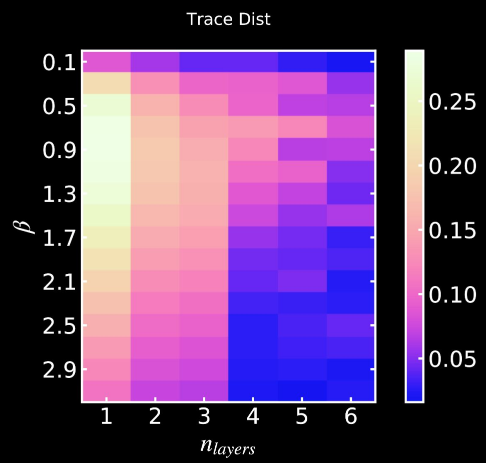
Nx = Ny = 2, at β = 2.6, Jx = 1.0, Jy = 0.6]

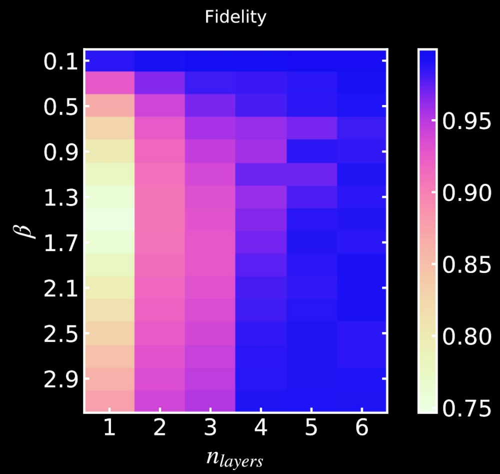
Variational Quantum Thermalization

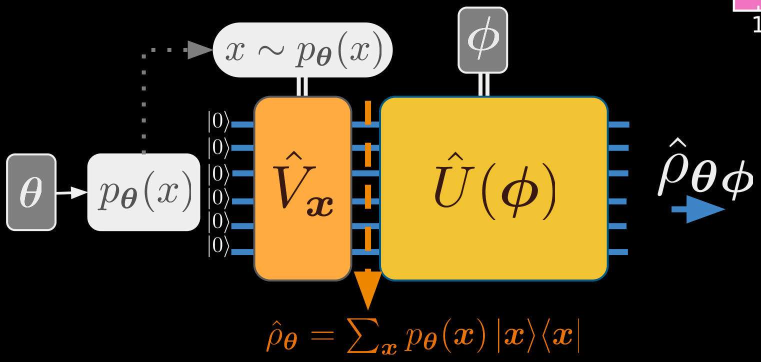
More recent Results

Sweeping over temperatures and QNN circuit depth for the Heisenberg model

Showing a relation between temperature, quantum correlations, and necessary quantum circuit depth







Final trace distance & fidelity after full VQT training

Quantum Variational Thermalization

Preparing Quantum Thermal States with Quantum-probabilistic inference

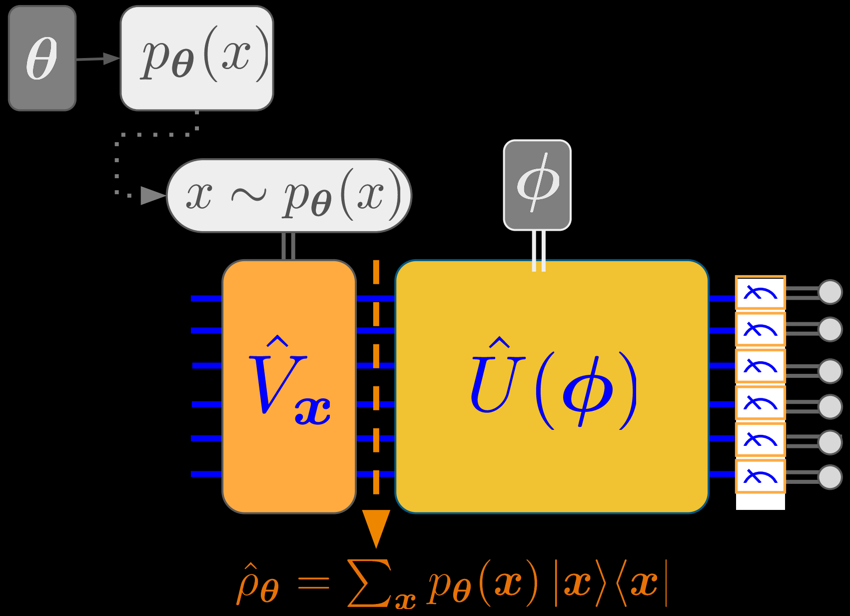
Quantum Simulation - the Variational Quantum Thermalizer (VQT)

Task: given target $\hat{H}$ and $\beta = 1/T$, generate $\hat{\sigma}_\beta = \frac{1}{\mathcal{Z}_\beta} e^{-\beta \hat{H}}, \quad \mathcal{Z}_\beta = \mathrm{tr}(e^{-\beta \hat{H}})$

Preparing Quantum Thermal States with Quantum-probabilistic inference

Quantum Simulation - the Variational Quantum Thermalizer (VQT)

Task: given target $\hat{H}$ and $\beta = 1/T$, generate $\hat{\sigma}_\beta = \frac{1}{\mathcal{Z}_\beta} e^{-\beta \hat{H}}, \quad \mathcal{Z}_\beta = \mathrm{tr}(e^{-\beta \hat{H}})$



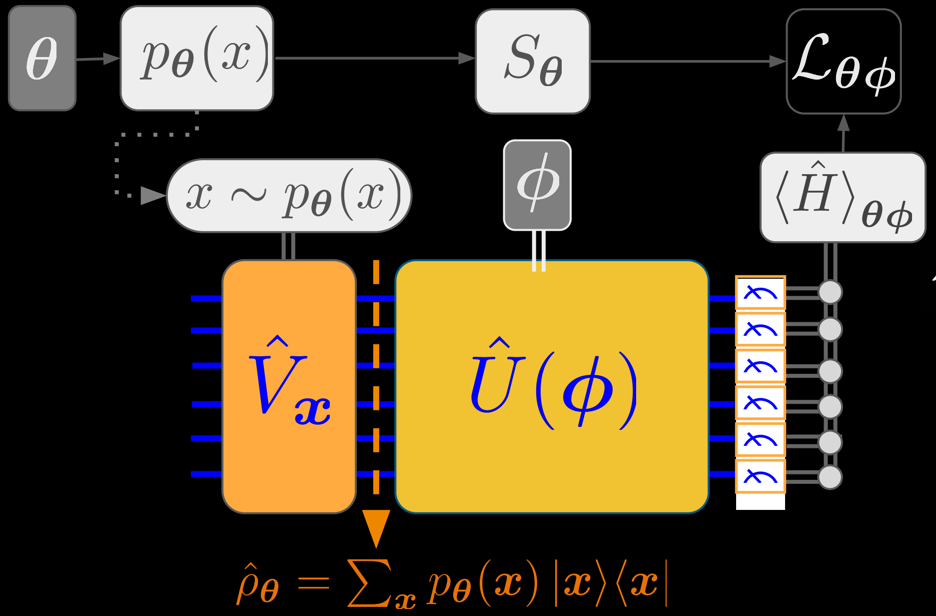
Variational hybrid model

$$\hat{\rho}_{\boldsymbol{\theta\phi}} = \hat{U}(\boldsymbol{\phi})\hat{\rho}_{\boldsymbol{\theta}}\hat{U}^\dagger(\boldsymbol{\phi})$$

Preparing Quantum Thermal States with Quantum-probabilistic inference

Quantum Simulation - the Variational Quantum Thermalizer (VQT)

Task: given target $\hat{H}$ and $\beta = 1/T$, generate $\hat{\sigma}_\beta = \frac{1}{\mathcal{Z}_\beta} e^{-\beta \hat{H}}, \quad \mathcal{Z}_\beta = \mathrm{tr}(e^{-\beta \hat{H}})$



Variational hybrid model

$$\hat{\rho}_{\boldsymbol{\theta}\boldsymbol{\phi}} = \hat{U}(\boldsymbol{\phi}) \hat{\rho}_{\boldsymbol{\theta}} \hat{U}^\dagger(\boldsymbol{\phi})$$

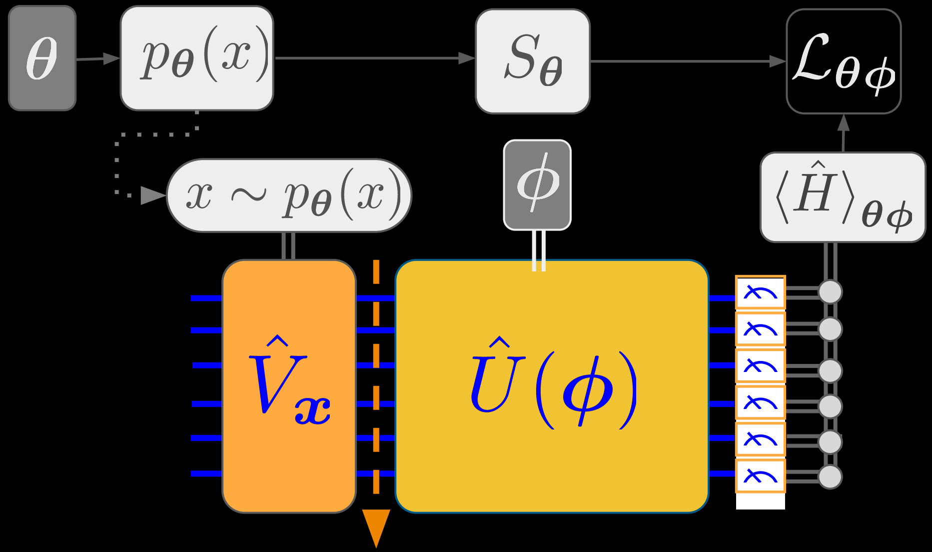
Minimize free energy

$$\mathcal{L}_{\mathrm{VQT}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \beta \mathrm{tr}(\hat{\rho}_{\boldsymbol{\theta}\boldsymbol{\phi}} \hat{H}) - S(\hat{\rho}_{\boldsymbol{\theta}\boldsymbol{\phi}})$$

Preparing Quantum Thermal States with Quantum-probabilistic inference

Quantum Simulation - the Variational Quantum Thermalizer (VQT)

Task: given target $\hat{H}$ and $\beta = 1/T$, generate $\hat{\sigma}_\beta = \frac{1}{\mathcal{Z}_\beta} e^{-\beta \hat{H}}, \quad \mathcal{Z}_\beta = \mathrm{tr}(e^{-\beta \hat{H}})$



$$\hat{\rho}_{\boldsymbol{\theta}} = \sum_{\boldsymbol{x}} p_{\boldsymbol{\theta}}(\boldsymbol{x}) \, |\boldsymbol{x}\rangle\langle\boldsymbol{x}|$$

Variational hybrid model

$$\hat{\rho}_{\boldsymbol{\theta}\boldsymbol{\phi}} = \hat{U}(\boldsymbol{\phi}) \hat{\rho}_{\boldsymbol{\theta}} \hat{U}^\dagger(\boldsymbol{\phi})$$

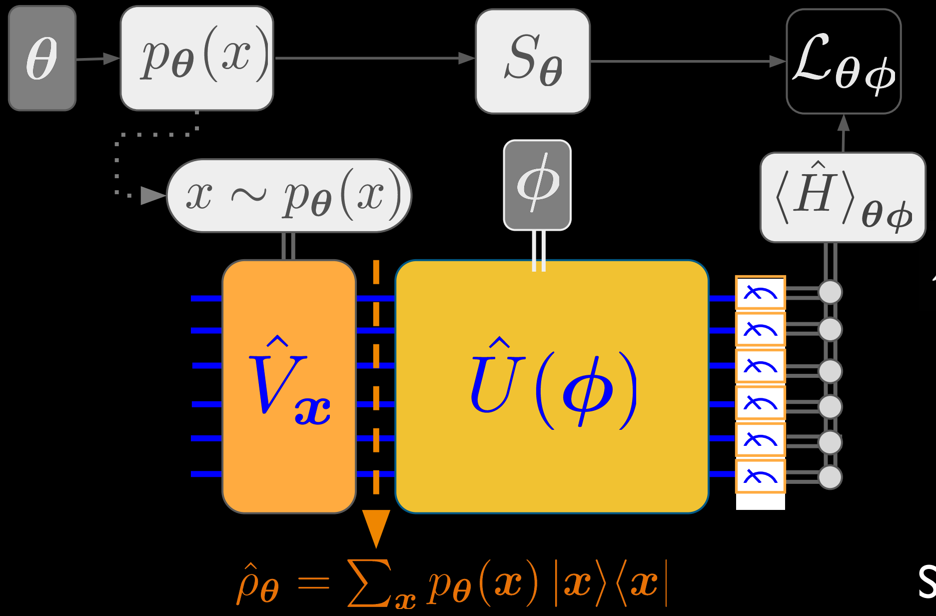
Minimize free energy

$$\mathcal{L}_{\mathrm{VQT}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \beta \mathrm{tr}(\hat{\rho}_{\boldsymbol{\theta}\boldsymbol{\phi}} \hat{H}) - S(\hat{\rho}_{\boldsymbol{\theta}\boldsymbol{\phi}})$$

Preparing Quantum Thermal States with Quantum-probabilistic inference

Quantum Simulation - the Variational Quantum Thermalizer (VQT)

Task: given target $\hat{H}$ and $\beta = 1/T$, generate $\hat{\sigma}_\beta = \frac{1}{\mathcal{Z}_\beta} e^{-\beta \hat{H}}, \quad \mathcal{Z}_\beta = \mathrm{tr}(e^{-\beta \hat{H}})$



Variational hybrid model

$$\hat{\rho}_{\boldsymbol{\theta\phi}} = \hat{U}(\boldsymbol{\phi})\hat{\rho}_{\boldsymbol{\theta}}\hat{U}^\dagger(\boldsymbol{\phi})$$

Minimize free energy

$$\mathcal{L}_{\mathrm{VQT}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \beta \mathrm{tr}(\hat{\rho}_{\boldsymbol{\theta\phi}} \hat{H}) - S(\hat{\rho}_{\boldsymbol{\theta\phi}})$$

Equivalent to finding

$$\underset{\boldsymbol{\theta},\boldsymbol{\phi}}{\mathrm{argmin}}\, D(\hat{\rho}_{\boldsymbol{\theta\phi}} \| \hat{\sigma}_\beta)$$

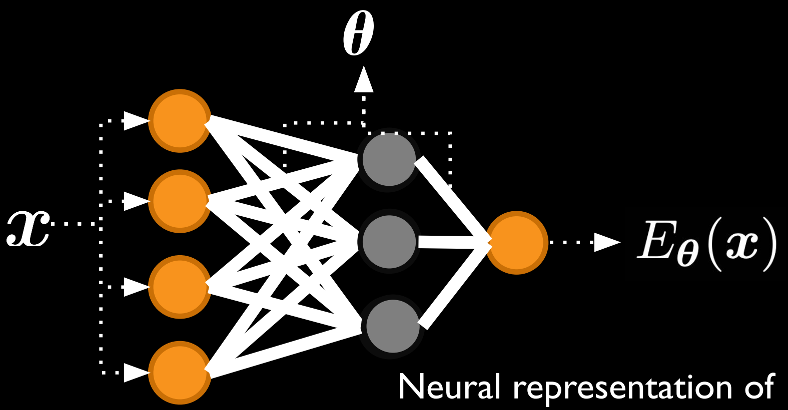
State of minimal free energy is thermal state!

Quantum Hamiltonian-Based Models

Combining classical probabilistic inference with quantum neural nets

Classical latent distribution parameterized by classical neural net

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{\mathcal{Z}_{\boldsymbol{\theta}}} e^{-E_{\boldsymbol{\theta}}(\boldsymbol{x})}$$



Neural representation of classical energy function

Latent Modular Hamiltonian representation

$$\hat{K}_{\boldsymbol{\theta}} = \sum_{\boldsymbol{x}\in\Omega} E_{\theta}(\boldsymbol{x}) |\boldsymbol{x}\rangle\langle\boldsymbol{x}|$$

$$\hat{\rho}_{\boldsymbol{\theta}} = \frac{1}{\mathcal{Z}_{\boldsymbol{\theta}}} e^{-\hat{K}_{\boldsymbol{\theta}}}, \qquad \mathcal{Z}_{\boldsymbol{\theta}} = \mathrm{tr}[e^{-\hat{K}_{\boldsymbol{\theta}}}]$$

Visible representation

$$\hat{\rho}_{\boldsymbol{\theta\phi}} = \frac{1}{\mathcal{Z}_{\boldsymbol{\theta}}} e^{-\hat{U}(\boldsymbol{\phi})\hat{K}_{\boldsymbol{\theta}}\hat{U}^{\dagger}(\boldsymbol{\phi})} \equiv \frac{1}{\mathcal{Z}_{\boldsymbol{\theta}}} e^{-\hat{K}_{\boldsymbol{\theta\phi}}}$$

Generative Learning of Quantum Mixed States with Quantum Hamiltonian-Based Models

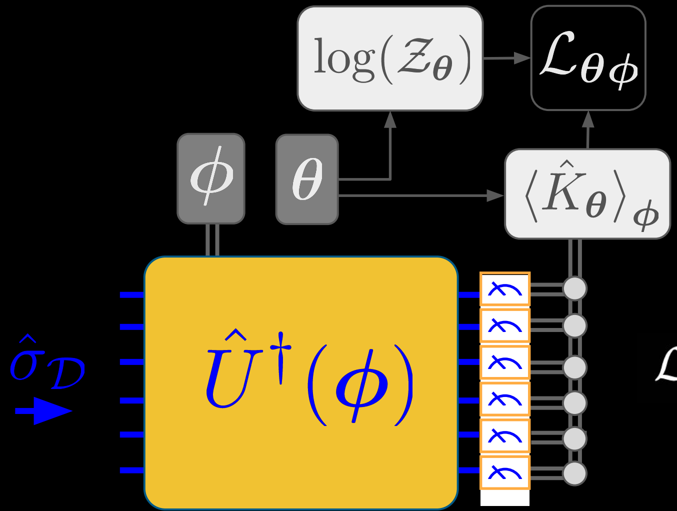
Quantum Modular Hamiltonian Learning for generative modelling

Task: given unknown $\hat{\sigma}_{\mathcal{D}} = \sum_{d\in\mathcal{D}} p_d\, \hat{\sigma}_d$ find $\{\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*\}$ such that $\hat{\rho}_{\boldsymbol{\theta}^*\boldsymbol{\phi}^*} \approx \hat{\sigma}_{\mathcal{D}}$

Generative Learning of Quantum Mixed States with Quantum Hamiltonian-Based Models

Quantum Modular Hamiltonian Learning for generative modelling

Task: given unknown $\hat{\sigma}_{\mathcal{D}} = \sum_{d\in\mathcal{D}} p_d\, \hat{\sigma}_d$ find $\{\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*\}$ such that $\hat{\rho}_{\boldsymbol{\theta}^*\boldsymbol{\phi}^*} \approx \hat{\sigma}_{\mathcal{D}}$



Approach:

Quantum Hamiltonian-Based Model:

$$\hat{\rho}_{\boldsymbol{\theta}\boldsymbol{\phi}} = \frac{1}{\mathcal{Z}_{\boldsymbol{\theta}}} e^{-\hat{U}(\boldsymbol{\phi})\hat{K}_{\boldsymbol{\theta}}\hat{U}^{\dagger}(\boldsymbol{\phi})} \equiv \frac{1}{\mathcal{Z}_{\boldsymbol{\theta}}} e^{-\hat{K}_{\boldsymbol{\theta}\boldsymbol{\phi}}}$$

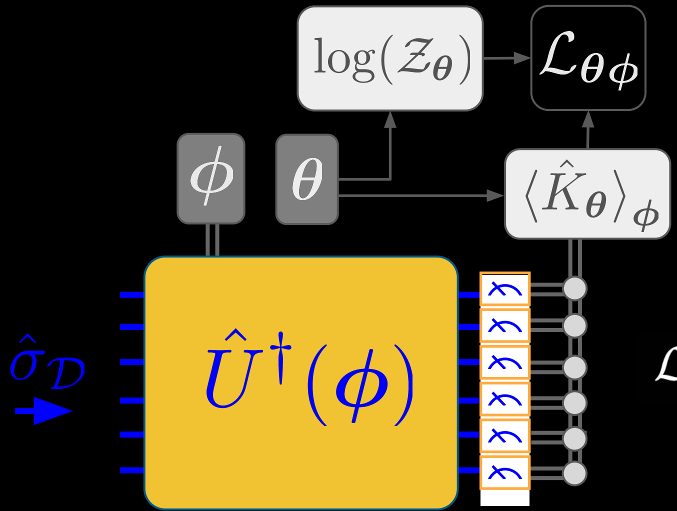
Minimize cross-entropy:

$$\mathcal{L}_{\text{QMHL}}(\boldsymbol{\theta}, \boldsymbol{\phi}) \equiv -\text{tr}(\hat{\sigma}_{\mathcal{D}} \log \hat{\rho}_{\boldsymbol{\theta}\boldsymbol{\phi}}) = \text{tr}(\hat{\sigma}_{\mathcal{D}} \hat{K}_{\boldsymbol{\theta}\boldsymbol{\phi}}) + \log(\mathcal{Z}_{\boldsymbol{\theta}})$$

Generative Learning of Quantum Mixed States with Quantum Hamiltonian-Based Models

Quantum Modular Hamiltonian Learning for generative modelling

Task: given unknown $\hat{\sigma}_{\mathcal{D}} = \sum_{d\in\mathcal{D}} p_d\, \hat{\sigma}_d$ find $\{\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*\}$ such that $\hat{\rho}_{\boldsymbol{\theta}^*\boldsymbol{\phi}^*} \approx \hat{\sigma}_{\mathcal{D}}$



Approach:

Quantum Hamiltonian-Based Model:

$$\hat{\rho}_{\boldsymbol{\theta\phi}} = \frac{1}{\mathcal{Z}_{\boldsymbol{\theta}}} e^{-\hat{U}(\boldsymbol{\phi})\hat{K}_{\boldsymbol{\theta}}\hat{U}^\dagger(\boldsymbol{\phi})} \equiv \frac{1}{\mathcal{Z}_{\boldsymbol{\theta}}} e^{-\hat{K}_{\boldsymbol{\theta\phi}}}$$

Minimize cross-entropy:

$$\mathcal{L}_{\mathrm{QMHL}}(\boldsymbol{\theta}, \boldsymbol{\phi}) \equiv -\mathrm{tr}(\hat{\sigma}_{\mathcal{D}} \log \hat{\rho}_{\boldsymbol{\theta\phi}}) = \mathrm{tr}(\hat{\sigma}_{\mathcal{D}} \hat{K}_{\boldsymbol{\theta\phi}}) + \log(\mathcal{Z}_{\boldsymbol{\theta}})$$

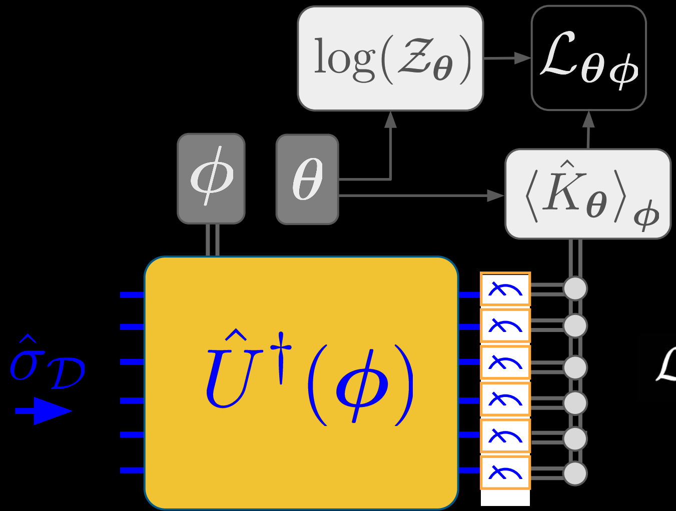
Equiv. to minimizing quantum relative entropy

$$\underset{\boldsymbol{\theta},\boldsymbol{\phi}}{\mathrm{argmin}}\, D(\hat{\sigma}_{\mathcal{D}} \| \hat{\rho}_{\boldsymbol{\theta\phi}})$$

Generative Learning of Quantum Mixed States with Quantum Hamiltonian-Based Models

Quantum Modular Hamiltonian Learning for generative modelling

Task: given unknown $\hat{\sigma}_{\mathcal{D}} = \sum_{d\in\mathcal{D}} p_d\, \hat{\sigma}_d$ find $\{\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*\}$ such that $\hat{\rho}_{\boldsymbol{\theta}^*\boldsymbol{\phi}^*} \approx \hat{\sigma}_{\mathcal{D}}$



Approach:

Quantum Hamiltonian-Based Model:

$$\hat{\rho}_{\boldsymbol{\theta}\boldsymbol{\phi}} = \frac{1}{\mathcal{Z}_{\boldsymbol{\theta}}} e^{-\hat{U}(\boldsymbol{\phi})\hat{K}_{\boldsymbol{\theta}}\hat{U}^\dagger(\boldsymbol{\phi})} \equiv \frac{1}{\mathcal{Z}_{\boldsymbol{\theta}}} e^{-\hat{K}_{\boldsymbol{\theta}\boldsymbol{\phi}}}$$

Minimize cross-entropy:

$$\mathcal{L}_{\text{QMHL}}(\boldsymbol{\theta}, \boldsymbol{\phi}) \equiv -\text{tr}(\hat{\sigma}_{\mathcal{D}} \log \hat{\rho}_{\boldsymbol{\theta}\boldsymbol{\phi}}) = \text{tr}(\hat{\sigma}_{\mathcal{D}} \hat{K}_{\boldsymbol{\theta}\boldsymbol{\phi}}) + \log(\mathcal{Z}_{\boldsymbol{\theta}})$$

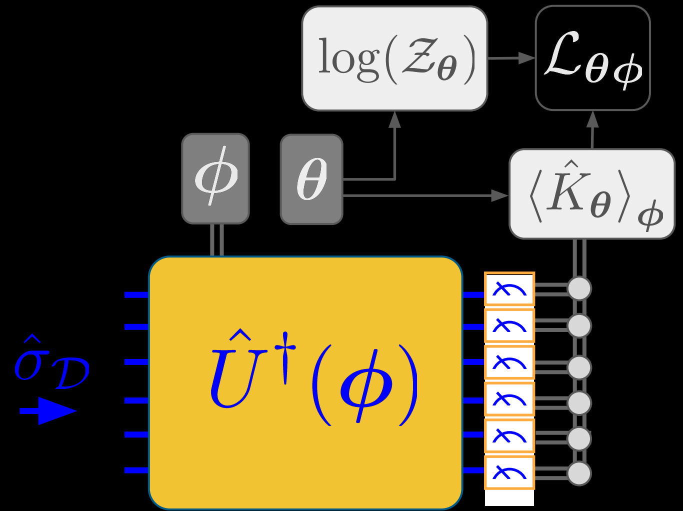
Cross entropy converges to entropy of data

$$\mathcal{L}_{\text{QMHL}}(\boldsymbol{\theta}, \boldsymbol{\phi}) \xrightarrow{\hat{\rho}_{\boldsymbol{\theta}\boldsymbol{\phi}} \to \hat{\sigma}_{\mathcal{D}}} S(\hat{\sigma}_{\mathcal{D}})$$

Generative Learning of Quantum Mixed States with Quantum Hamiltonian-Based Models

Quantum Modular Hamiltonian Learning for generative modelling

Gradients? $\mathcal{L}_{\text{QMHL}}(\boldsymbol{\theta}, \boldsymbol{\phi}) \equiv -\text{tr}(\hat{\sigma}_{\mathcal{D}} \log \hat{\rho}_{\boldsymbol{\theta\phi}}) = \text{tr}(\hat{\sigma}_{\mathcal{D}} \hat{K}_{\boldsymbol{\theta\phi}}) + \log(\mathcal{Z}_{\boldsymbol{\theta}})$



Classical Model Gradients

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{QMHL}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \nabla_{\boldsymbol{\theta}} \text{tr}([\hat{U}^\dagger(\boldsymbol{\phi}) \hat{\sigma}_{\mathcal{D}} \hat{U}(\boldsymbol{\phi})] \hat{K}_{\boldsymbol{\theta}}) + \nabla_{\boldsymbol{\theta}} \log(\mathcal{Z}_{\boldsymbol{\theta}})$$

$$= \mathbb{E}_{\boldsymbol{x} \sim \sigma_{\boldsymbol{\phi}}(\boldsymbol{x})} [\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{y} \sim p_{\theta}(\boldsymbol{y})} [\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{y})]$$

$$\text{where } \sigma_{\boldsymbol{\phi}}(\boldsymbol{x}) \equiv \langle \boldsymbol{x} | \hat{U}^\dagger(\boldsymbol{\phi}) \hat{\sigma}_{\mathcal{D}} \hat{U}(\boldsymbol{\phi}) | \boldsymbol{x} \rangle$$

QNN gradients: parameter shift on reverse unitary