

Learning Fidelity Susceptibility from Quantum-Simulator Snapshots

*A mini project by Jonas Rigo¹ for the
Quantum ides factory challenge*

Abstract

Quantum simulators (QS), such as programmable Rydberg-atom arrays, enable precise realizations of many-body Hamiltonians with highly tunable control parameters, making it possible to scan phase diagrams in a controlled and systematic way. One of the most direct observables provided by these platforms are *occupation snapshots*: repeated measurements in the occupation basis that yield bitstring samples distributed according to the Born rule of the prepared quantum state. Because such snapshots do not contain information about the complex phase of the wave function, the full quantum state cannot be reconstructed from single-basis data alone, which can complicate the detection of phase transitions. Nevertheless, a machine-learning model can learn the Born distribution $p(s | \lambda)$ as a function of a Hamiltonian parameter λ , provided sufficient snapshot data are available at each parameter setting. Once trained, the parameter-dependent model enables the evaluation of a *classical proxy for the fidelity susceptibility* based only on probability distributions. The fidelity susceptibility is an essentially system-agnostic diagnostic that is particularly useful for identifying phase transitions. In this project, we train such a model on occupation snapshots of the one-dimensional spinless t - V model and compute the classical fidelity susceptibility across a sweep of its phase diagram.

LITERATURE BACKGROUND

This document, along with the GitHub repository, provides all the information necessary to implement the mini-project. For additional technical background and a discussion of the relationship between QS and classical learning algorithms, we refer you to the following excellent readings:

- *Transformer Neural Networks and Quantum Simulators: A Hybrid Approach for Simulating Strongly Correlated Systems* [1]
- *Efficient Quantum State Tomography with Convolutional Neural Networks* [2]
- *Recurrent Neural Network Wave Functions* [3]
- *Fidelity Susceptibility Made Simple: A Unified Quantum Monte Carlo Approach* [4]
- *Foundation Neural-Networks Quantum States as a Unified Ansatz for Multiple Hamiltonians* [5]

I. PROJECT GOALS

1. What you will build

A conditional generative model for occupation basis snapshots,

$$p_\theta(s | \lambda) \approx p_{\text{QS}}(s | \lambda), \quad (1)$$

trained by maximum cross-entropy on snapshot datasets. Using the trained model, you will estimate the Bhattacharyya distance (a proxy for the *fidelity*) between neighboring parameters and form a classical fidelity susceptibility $\chi_B(\lambda)$ that peaks near the phase-transition region.

¹ jonas.rigo@proton.me

2. What you will learn

- How snapshot measurements from quantum simulators relate to Born distributions.
- How conditional generative models learn $p(s | \lambda)$ from data using cross-entropy.
- How to use the fidelity susceptibility to detect phase transitions
- How autoregressive (RNN) models enable exact likelihood evaluation and sampling.
- The 1D spinless $t-V$ model and its *charge-density wave* (CDW) transition, including an order parameter you can compute from occupation snapshots.

3. What you should produce

1. A routine that loads the sample sets of occupation snapshots and compute the order parameter for each λ .
2. A RNN with sample and cross-entropy evaluation routines is provided, you need to write a training routine for the RNN, a routine to compute the fidelity susceptibility and the CDW order parameter. conditional autoregressive network $p_\theta(s | \lambda)$ trained on mixed- λ batches.
3. Plots: (i) training/validation NLL vs epoch, (ii) CDW order parameter vs λ and its derivative with respect to λ , (iii) $\chi_B(\lambda)$ vs λ with a peak.

II. QUANTUM SIMULATORS, NEURAL IMPLICIT QUANTUM STATES, AND SNAPSHOTS

A. Quantum simulators and measurement snapshots

A quantum simulator is a controllable quantum device engineered to realize a target Hamiltonian $H(\lambda)$ and prepare its states (often ground states or low-energy states) by adiabatic ramps or variational protocols. In *Rydberg-atom arrays*, neutral atoms trapped in optical tweezers can be arranged in programmable geometries with adjustable interaction strengths. After preparing the state, the system is measured repeatedly in a fixed basis, producing a dataset of *occupation states*. If the prepared state at parameter λ is $|\psi(\lambda)\rangle$, and one measures in the computational/occupation basis $\{|s\rangle\}$, then each repetition yields a classical outcome s distributed as

$$p_{\text{QS}}(s | \lambda) = |\langle s | \psi(\lambda) \rangle|^2. \quad (2)$$

Collecting many repetitions gives a snapshot dataset $\{s^{(k)}\}_{k=1}^N$ drawn *independently identically distributed* (i.i.d.) from $p_{\text{QS}}(s | \lambda)$ (up to experimental noise and state-preparation imperfections).

B. Modeling the state implicitly via probabilities

A *neural quantum state* (NQS) usually models complex amplitudes $\psi_\theta(s)$ directly. In this project we focus on the *implicit* description, that is we model only the *Born probabilities*

$$p_\theta(s | \lambda) \approx |\psi_\theta(s | \lambda)|^2, \quad (3)$$

learned from snapshot data in a single measurement basis. This is sufficient to compute any observable diagonal in that basis, and it enables phase-transition detection via distributional changes. It is important to note that, single-basis snapshots do not *uniquely* determine wave-function phases, but with the knowledge of the Hamiltonian the phase could technically be computed (although in a very expensive process).

III. LEARNING PROBABILITIES WITH A PARAMETER-DEPENDENT MODEL

A. Conditional likelihood and cross-entropy loss

We train a neural network with parameters θ that outputs a normalized probability distribution $p_\theta(s \mid \lambda)$ for occupation states s given a control parameter λ (e.g. $\lambda = V/t$). Given a dataset $\mathcal{D} = \{(s^{(k)}, \lambda^{(k)})\}$, maximum likelihood training minimizes the negative log-likelihood (NLL)

$$\mathcal{L}_{\text{NLL}}(\theta) = -\frac{1}{|\mathcal{D}|} \sum_{(s, \lambda) \in \mathcal{D}} \log p_\theta(s \mid \lambda). \quad (4)$$

For a fixed λ , this is the cross-entropy between the empirical distribution $\hat{p}_{\text{data}}(\cdot \mid \lambda)$ and the model:

$$\mathcal{L}_{\text{NLL}}(\theta; \lambda) = -\sum_s \hat{p}_{\text{data}}(s \mid \lambda) \log p_\theta(s \mid \lambda). \quad (5)$$

The lower the NLL, the higher the probability the model assigns (on average) to typical snapshots from the data distribution, and therefore the better the fit.

B. Teacher Forcing training

For an autoregressive model,

$$p_\theta(s \mid \lambda) = \prod_{i=1}^L p_\theta(s_i \mid s_{<i}, \lambda), \quad s_{<i} = (s_1, \dots, s_{i-1}). \quad (6)$$

The NLL decomposes into a sum of per-step cross-entropies:

$$-\log p_\theta(s \mid \lambda) = -\sum_{i=1}^L \log p_\theta(s_i \mid s_{<i}, \lambda). \quad (7)$$

Teacher forcing refers to feeding the *ground-truth prefix* $s_{<i}$ as the network input when predicting s_i . With teacher forcing, the NLL in Eq. (4) is evaluated exactly and efficiently. The procedure is to shift the sequence by one (prepend a start token) and train the model to predict the next token.

C. Scheduled sampling (training-time modification)

Teacher forcing trains on prefixes drawn from the data distribution. During generation, however, the model conditions on its *own* previously generated tokens. This mismatch can cause *exposure bias*: small early mistakes lead to unrealistic prefixes at later steps.

Scheduled sampling addresses this by occasionally replacing the teacher-forced input token with a token generated by the model itself, while still scoring the loss on the true next token. Concretely, at step i we construct the input token x_{i-1} via a Bernoulli switch (probabilistic binary choice),

$$x_{i-1} = \begin{cases} s_{i-1}, & \text{with probability } \varepsilon \\ \tilde{s}_{i-1}, & \text{with probability } 1 - \varepsilon, \quad \tilde{s}_{i-1} \sim p_\theta(\cdot \mid x_{<i-1}, \lambda), \end{cases} \quad (8)$$

and minimize

$$\mathcal{L}_{\text{SS}}(\theta) = -\mathbb{E}_{(s, \lambda) \sim \text{data}} \left[\sum_{i=1}^L \log p_\theta(s_i \mid x_{<i}, \lambda) \right]. \quad (9)$$

Here $\varepsilon \in [0, 1]$ is the *teacher probability*. The baseline teacher-forcing regime is recovered at $\varepsilon = 1$. In practice, ε is annealed from $\varepsilon_{\text{max}} \approx 1$ to a moderate ε_{min} (e.g. 0.7) using a smooth schedule (cosine annealing is a good default).

Even if scheduled sampling is used for training, validation is typically performed with *pure teacher forcing* that is, the standard NLL of Eq. (4). This yields a stable, comparable metric across runs. Generation quality is then assessed separately using observables computed from model-generated samples.

IV. FIDELITY SUSCEPTIBILITY AND THE CLASSICAL (BHATTACHARYYA) PROXY

A. Quantum fidelity susceptibility (context)

For two *pure quantum states* $|\psi(\lambda)\rangle$ and $|\psi(\lambda + \delta)\rangle$, the quantum fidelity is

$$F(\lambda, \lambda + \delta) = |\langle\psi(\lambda)|\psi(\lambda + \delta)\rangle|^2. \quad (10)$$

The fidelity susceptibility is defined from the small- δ expansion

$$F(\lambda, \lambda + \delta) \approx 1 - \frac{\chi_F(\lambda)}{2}\delta^2 + \mathcal{O}(\delta^3), \quad (11)$$

and often peaks near phase transitions. However, computing F requires wave-function phases, which are not available from single-basis snapshots.

B. Classical fidelity susceptibility

Snapshots give access to probability distributions $p(s | \lambda)$ in a fixed basis. A natural overlap between classical distributions is the Bhattacharyya coefficient

$$B(\lambda, \lambda + \delta) = \sum_s \sqrt{p(s | \lambda)p(s | \lambda + \delta)} \in [0, 1]. \quad (12)$$

It satisfies $B(\lambda, \lambda) = 1$ and decreases as the distributions become more distinguishable. Because $B(\lambda, \lambda)$ is maximal at $\delta = 0$, the leading change is quadratic:

$$B(\lambda, \lambda + \delta) \approx \exp\left(-\frac{\chi_B(\lambda)}{2}\delta^2\right) \Rightarrow \chi_B(\lambda) \approx -\frac{2}{\delta^2} \log B(\lambda, \lambda + \delta). \quad (13)$$

Thus, although χ_B is defined from the curvature at $\delta \rightarrow 0$, in practice we estimate it using a finite (small) step δ equal to the parameter grid spacing.

C. Stochastic estimator for B using a generative model

If we can evaluate normalized probabilities $p_\theta(s | \lambda)$ and sample from them, we can estimate B without summing over all s . Using importance sampling with $s \sim p_\theta(\cdot | \lambda)$,

$$B(\lambda, \lambda + \delta) = \mathbb{E}_{s \sim p(\cdot | \lambda)} \left[\sqrt{\frac{p(s | \lambda + \delta)}{p(s | \lambda)}} \right]. \quad (14)$$

A symmetric estimator (recommended for variance reduction) averages both directions:

$$B(\lambda, \lambda + \delta) \approx \frac{1}{2} \mathbb{E}_{s \sim p(\cdot | \lambda)} \left[\sqrt{\frac{p(s | \lambda + \delta)}{p(s | \lambda)}} \right] + \frac{1}{2} \mathbb{E}_{s \sim p(\cdot | \lambda + \delta)} \left[\sqrt{\frac{p(s | \lambda)}{p(s | \lambda + \delta)}} \right]. \quad (15)$$

You will use p_θ in place of the unknown true p .

V. GENERATIVE MODELS

A *generative model* earns its name from the ability to generate samples from its captured probability distribution $s \sim p_\theta(\cdot | \lambda)$. For generative *autoregressive* models, the sampling process is best understood by breaking down the events s into a sequence: Let $s = (s_1, \dots, s_L)$ be a length- L bitstring (or small alphabet). An autoregressive model uses the exact factorization

$$p_\theta(s | \lambda) = \prod_{i=1}^L p_\theta(s_i | s_{<i}, \lambda), \quad (16)$$

where $s_{<i} = (s_1, \dots, s_{i-1})$ is the prefix. A recurrent neural network (RNN) is a specific type of autoregressive network, it processes the sequence one symbol at a time while maintaining a hidden “memory” state h_i that summarizes the prefix. At each step it outputs logits $\ell_i \sim \log p_\theta(s_i | s_{<i}, \lambda)$ for the next symbol. More about RNNs in quantum many-body physics can be found in Ref [3]. It is important that RNNs allow us to:

- *Compute exact likelihoods:* Given a full sequence s , the network produces all conditional probabilities in (16), allowing exact evaluation of $\log p_\theta(s | \lambda)$ by summing $\log p_\theta(s_i | s_{<i}, \lambda)$ over i .
- *Sample autoregressively:* To sample, start from an empty prefix and repeatedly draw

$$s_i \sim p_\theta(\cdot | s_{<i}, \lambda), \quad (17)$$

feeding each sampled symbol back into the RNN. This produces independent samples directly from the model distribution (no Markov-chain mixing required), making it ideal for laptops.

VI. THE 1D SPINLESS t - V MODEL AND THE CDW TRANSITION

We consider spinless fermions on a 1D ring (periodic boundary conditions or PBCs) with Hamiltonian

$$H(\lambda) = -t \sum_{j=1}^L (c_j^\dagger c_{j+1} + c_{j+1}^\dagger c_j) + V \sum_{j=1}^L n_j n_{j+1}, \quad \lambda \equiv V/t, \quad (18)$$

where $c_{L+1} \equiv c_1$ under PBC, and $n_j = c_j^\dagger c_j$. We focus on half filling, $N_f = L/2$, which can be set by adjusting an additional chemical potential or projecting $H(\lambda)$ into the right occupation space. Note that this is already implicitly done in the the occupation basis states obtained from the QS.

Our interest in this model is the aspect taht at small $\lambda = V/t$, the system is a gapless Luttinger liquid with strong quantum fluctuations (the 1D equivalent of a Fermi liquid). As λ increases, repulsion favors alternating density patterns. In the thermodynamic limit, a transition occurs near $\lambda_c = 2$ between the Luttinger liquid and a charge-density wave ordered phase. On finite $L \sim 12$, this appears as a smooth crossover, which is sufficient for this project. To visualize the crossover a convenient CDW diagnostic is the structure factor at momentum $q = \pi$:

$$S(\pi) = \frac{1}{L} \sum_{j,k} (-1)^{j-k} \langle (n_j - \bar{n})(n_k - \bar{n}) \rangle, \quad \bar{n} = \frac{N_f}{L} = \frac{1}{2}. \quad (19)$$

This is diagonal in the occupation basis and can be estimated directly from snapshots (either QS snapshots or samples from p_θ). As λ increases past the transition region, $S(\pi)$ grows markedly.

VII. PRACTICAL WORKFLOW AND CHECKLIST

A. Data import

1. Choose system size $L \in \{10, 12, 14\}$ and load the respective QS samples file.
2. Verify the expected physical behavior of the t - V model by computing the structure factor at momentum $q = \pi$.

B. Model training

1. Train a conditional autoregressive model $p_\theta(s | \lambda)$ by minimizing NLL (4) (teacher-forced evaluation).
2. Mix/shuffle λ values during training with balanced sampling.
3. Monitor validation NLL per λ .

4. (**PBC data augmentation: random translations**). With periodic boundary conditions, the physics is translation invariant. Enforce this in training by randomly translating each snapshot by a uniformly random shift $r \in \{0, \dots, L - 1\}$:

$$s \mapsto \text{roll}(s, r).$$

In code, this is simply `s = np.roll(s, shift=r, axis=-1)` (and similarly for model-generated samples when comparing observables). This augmentation greatly improves sample efficiency and helps the network learn translation-invariant structure.

C. Compute $\chi_B(\lambda)$

1. For each grid point λ_m , estimate $B(\lambda_m, \lambda_m + \delta)$ using (15) with samples from the trained model at λ_m and $\lambda_m + \delta$.
2. Compute

$$\chi_B(\lambda_m) \approx -\frac{2}{\delta^2} \log B(\lambda_m, \lambda_m + \delta).$$

3. Plot $\chi_B(\lambda)$ and verify a peak near the CDW transition region.

D. Sanity checks

- Compare $S(\pi)$ computed from model samples against ED ground truth (should match within error bars if training succeeded).
- Verify that $\chi_B(\lambda)$ peak roughly coincides with the region where $S(\pi)$ grows fastest (you can compute the finite difference derivative of $S(\pi)$ to find that region)
- Check stability with respect to sample count and step size δ (smaller δ is more local but noisier).

VIII. BONUS

- **Finite-size scaling:** repeat for $L = 10, 12, 14$ and track the shift/narrowing of the χ_B peak.
- **Snapshot noise:** add bit-flip noise to snapshots (mimicking readout errors) and study robustness of χ_B .

- [1] H. Lange, G. Bornet, G. Emperauger, C. Chen, T. Lahaye, S. Kienle, A. Browaeys, and A. Bohrdt, Transformer neural networks and quantum simulators: A hybrid approach for simulating strongly correlated systems, *Quantum* **9**, 1675 (2025), arXiv:2406.00091 [cond-mat].
- [2] T. Schmale, M. Reh, and M. Gärttner, Efficient quantum state tomography with convolutional neural networks, *npj Quantum Information* **8**, 115 (2022).
- [3] M. Hibat-Allah, M. Ganahl, L. E. Hayward, R. G. Melko, and J. Carrasquilla, Recurrent neural network wave functions, *Physical Review Research* **2**, 023358 (2020).
- [4] L. Wang, Y.-H. Liu, J. Imriška, P. N. Ma, and M. Troyer, Fidelity susceptibility made simple: A unified quantum Monte Carlo approach, *Physical Review X* **5**, 031007 (2015), arXiv:1502.06969 [cond-mat].
- [5] R. Rende, L. L. Viteritti, F. Becca, A. Scardicchio, A. Laio, and G. Carleo, Foundation neural-networks quantum states as a unified Ansatz for multiple hamiltonians, *Nature Communications* **16**, 7213 (2025).