

# Meta Hamiltonian Learning

Przemyslaw Bienias,<sup>1,2,\*</sup> Alireza Seif,<sup>3,\*</sup> and Mohammad Hafezi<sup>1,4</sup>

<sup>1</sup>Joint Quantum Institute, NIST/University of Maryland, College Park, Maryland 20742 USA

<sup>2</sup>Joint Center for Quantum Information and Computer Science,

NIST/University of Maryland, College Park, Maryland 20742 USA

<sup>3</sup>Pritzker School of Molecular Engineering, University of Chicago, Chicago, Illinois 60637, USA

<sup>4</sup>Department of Electrical and Computer Engineering and Institute for Research in Electronics and Applied Physics, University of Maryland, College Park, Maryland 20742, USA

(Dated: April 12, 2021)

Efficient characterization of quantum devices is a significant challenge critical for the development of large scale quantum computers. We consider an experimentally motivated situation, in which we have a decent estimate of the Hamiltonian, and its parameters need to be characterized and fine-tuned frequently to combat drifting experimental variables. We use a machine learning technique known as meta-learning to learn a more efficient optimizer for this task. We consider training with the nearest-neighbor Ising model and study the trained model’s generalizability to other Hamiltonian models and larger system sizes. We observe that the meta-optimizer outperforms other optimization methods in average loss over test samples. This advantage follows from the meta-optimizer being less likely to get stuck in local minima, which highly skews the distribution of the final loss of the other optimizers. In general, meta-learning decreases the number of calls to the experiment and reduces the needed classical computational resources.

*Introduction.*—Recently, there has been significant progress in experimental realizations of quantum computers [1–3] and quantum simulators [4]. This progress further motivates the need for characterization and validation of quantum devices, which is crucial for their precise control and operation. Resources required for complete characterization of a quantum state or a quantum process, known as quantum tomography, scales exponentially with the system size [5]. Specifically, learning the Hamiltonian of a quantum system is one of the extensively studied directions in the field of quantum characterization [5, 6]. There are different approaches to address this problem more efficiently: linear inversion based tomographic methods and compressed sensing [7–10], maximum likelihood optimization or Bayesian methods [11–16], system identification algorithms [17–20], and techniques that exploit physical knowledge about the system to design model specific measurements to learn the Hamiltonian parameters [21–25]. In addition to the above techniques, machine learning methods have been used to characterize quantum states and quantum processes [26–28].

Moreover, in noisy intermediate-scale quantum devices, control is limited, and it is not possible to prepare and measure many different configurations as required in many Hamiltonian estimation techniques. Current approaches make use of time-traces of observables with a few easy-to-prepare initial states to estimate parameters of a model Hamiltonian by optimizing a cost function that quantifies how well a model’s predictions agree with the observed measurement outcomes [15]. Unfortunately, this fitting procedure requires a classical simulation of the quantum dynamics that is inherently difficult. Therefore, it is interesting to investigate whether machine learning tools can help with accelerating this process. Here, we

use a machine learning technique known as meta-learning (also named as *learning to learn*) [29], to design an algorithm that learns how to more efficiently optimize the cost function. This is achieved through training the meta-optimizer by solving samples of the optimization problem of interest.

Remarkably, we find that the meta-optimizer algorithm (Fig. 1) performs better than the other optimizers we considered (Fig. 2). Specifically, it achieves the lowest mean loss (0.006), a 6-fold decrease compared to the next best optimizer (0.040), over 300 test instances. Equivalently, the meta-optimizer achieves the same loss as the next best optimizer with fewer number of iterations (16 versus 100). Moreover, the spread of the test loss values characterized by the standard deviation is an order of magnitude smaller for the meta-optimizer (0.099) compared to L-BFGS (1.080). As the data is highly skewed, we consider more detailed statistics later in this Letter. An important feature of our meta-optimizer is its flexibility in the number of input variables. We show that the algorithm successfully generalizes to both larger systems within the same class of models but with more parameters (Fig. 3), and to different classes of models (Fig. 4). We also demonstrate that the meta-optimizer is robust against noise in all cases (Fig. 5).

*System.*—We consider the task of inferring the Hamiltonian of the system from time traces of some observables in the experiment. Specifically, we model the system’s Hamiltonian with  $H(\boldsymbol{\theta}) = \sum_i \theta_i P_i$ , where  $\boldsymbol{\theta}$  is a vector that contains the parameters of the model  $\theta_i$ ’s, and  $P_i$ ’s are operators in the system’s Hilbert space. Note that in general, any Hamiltonian for a system of qubits can be written in this form if  $P_i$ ’s form a complete basis of operators in that Hilbert space, e.g., the Pauli basis. However, this set can be much more restricted due to

geometric considerations and physical constraints on the nature of the couplings. The system is first prepared in different initial states  $\{\rho_j\}$ . Then, the system evolves and the expectation value of a set of observables  $\{O_i\}$  is measured at different times for each initial state, that is  $y_{i,j}(t) = \text{tr}(\rho_j(t)O_i)$ . We then minimize the least-square loss

$$f(\boldsymbol{\theta}) = \sum_{i,j,t} (y_{i,j}(t) - \tilde{y}_{i,j}(t; \boldsymbol{\theta}))^2, \quad (1)$$

where  $\tilde{y}_{i,j}(t; \boldsymbol{\theta}) = \text{tr}\{\exp[-iH(\boldsymbol{\theta})t]\rho_j(0)\exp[iH(\boldsymbol{\theta})t]O_i\}$  is our model's prediction for  $y_{i,j}(t)$ . The resulting optimum,  $\hat{\boldsymbol{\theta}} = \text{argmin}_{\boldsymbol{\theta}} f(\boldsymbol{\theta})$ , is our estimate for the parameters of the model [30].

Obviously, solving such an optimization problem is complicated, and we may not be able to find global minima under general circumstances. However, we focus on the case of tuning parameters, and assume that we have a close guess of the value of the parameters. For example, in an ion-trap experiment [31] based on independently-measured sideband frequencies, one has a good estimate of the coupling strengths in the effective Ising Hamiltonian that governs the system. We should also note that a similar gradient-based optimization approach to Hamiltonian estimation has been previously considered in Ref. [15] by measuring the system's response to a control Hamiltonian. In contrast, we consider the response of the system to multiple initial states, which might be simpler in an experiment.

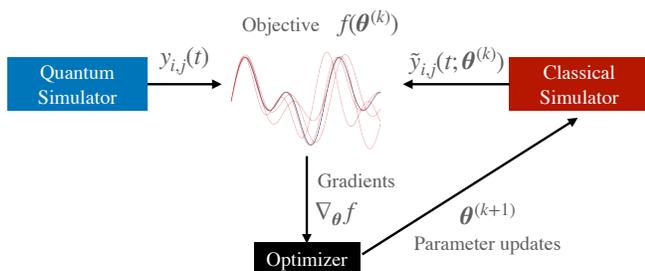


FIG. 1. *Schematic representation of the methods:* The measurement signal  $y_{i,j}(t)$  from the quantum device (blue) and our predicted signal  $\tilde{y}_{i,j}(t; \boldsymbol{\theta}^{(k)})$  given the current estimate of the model parameters  $\boldsymbol{\theta}^{(k)}$  (red) are compared to calculate the objective function  $f(\boldsymbol{\theta}^{(k)})$ . The gradient of the objective function,  $\nabla_{\boldsymbol{\theta}} f$ , is then provided to the optimizer, which updates our estimate for the parameters  $\boldsymbol{\theta}^{(k+1)}$ . This process is iterated to move model's prediction closer to the experimental observation.

*Meta-learning.*—In the machine learning context, learned representations and features often result in a better performance than what can be obtained with hand-designed representations and features [32]. Still, optimization algorithms themselves are mostly designed by

hand. However, an optimization algorithm's design can be formulated as a learning problem, so-called meta-learned optimization. It has been shown that meta-learned optimization outperforms generic hand-designed competitors in many tasks [29]. The application of this concept to the quantum domain has been recently considered in the context of the variational quantum algorithms [33, 34].

To understand the concept of meta-learning, it is worth comparing it to the conventional optimization algorithms. For example, consider gradient descent based optimizers for a differentiable cost function  $f$ . The parameters  $\boldsymbol{\theta}$  are updated in each optimization step  $k$  via  $\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \alpha_k \nabla f(\boldsymbol{\theta}^{(k)})$ , where  $\alpha_k$  is the learning rate at step  $k$ . In the meta-learning approach, however, the idea is to replace the hand-designed update rules with a learned update rule,  $\mathbf{g}_k$ , parameterized by  $\phi$ , such that

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \mathbf{g}_k(\nabla f(\boldsymbol{\theta}^{(k)}), \phi). \quad (2)$$

We explicitly used the subscript  $k$  in  $\mathbf{g}_k$ , to indicate that the update rule can depend on the optimization step  $k$ . We follow Ref. [29]'s approach and model the updates as the output of a recurrent neural network (RNN). The step dependence of the updates are implemented through the dynamically updated hidden state of the RNN,  $\mathbf{h}_k$ . Therefore,  $\mathbf{g}_k$  is explicitly given by  $[\mathbf{g}_k, \mathbf{h}_{k+1}] = m(\nabla f(\boldsymbol{\theta}^{(k)}), \mathbf{h}_k, \phi)$ , where  $m$  is the recurrent neural network function—referred to as the meta-optimizer—and  $\phi$  encodes the trainable parameters of the RNN. Specifically, we use a Long Short-Term Memory (LSTM) neural network, which is highly effective in processing sequential data. Compared to common RNNs, LSTM have the ability to forget and add new information as time runs, which solves the vanishing gradient problem encountered in training other RNNs [30, 35].

The performance of the optimizer is quantified with the expected loss,  $\mathbb{E}_f[f(\hat{\boldsymbol{\theta}})]$ , over a distribution of functions  $f$ . To train the meta-optimizer we can directly optimize the expected loss. However, it has been shown [29] that for the purpose of training the meta-optimizer, it is advantageous to instead minimize

$$\mathcal{L}(\phi) = \mathbb{E}_f \left[ \sum_{k=1}^T f(\boldsymbol{\theta}^{(k)}) \right], \quad (3)$$

that is the expected loss over optimization trajectories with the total of  $T$  steps.

As noted earlier, to tune and characterize the Hamiltonian parameters of a quantum system, one approach is to minimize Eq. (1) given estimates of  $y_{i,j}(t)$  obtained from the experiment. Such an optimization-based approach for estimating Hamiltonian parameters fits nicely in the meta-learning framework. Hence, we take advantage of automatic differentiation techniques that allows us to readily calculate  $\nabla_{\phi} \mathcal{L}$  and train a meta-optimizer for Hamiltonian learning.

*Setup.*—The meta-optimizer  $m$  is first trained on a system described by the nearest-neighbor Ising model. We then study the trained model’s generalizability under various circumstances, such as systems with more spins, different Hamiltonians, and larger noise in the input.

The training data is generated by numerically evaluating the quench dynamics in the nearest-neighbor transverse field Ising model (TFIM) with the Hamiltonian

$$H = \sum_i^N J_i X_i X_{i+1} + \sum_i^N B_i Z_i, \quad (4)$$

consisting of  $N = 4$  qubits with periodic boundary conditions. We use the convention  $\theta = \{J_1, \dots, J_N, B_1, \dots, B_N\}$ . We consider two easy to prepare initial states  $\rho_j(t=0)$  with  $j \in \{X, Z\}$ , corresponding to the state with all qubits aligned along either  $X$  or  $Z$  directions, respectively [30]. The training time series data  $y_{i,j}(t)$  are the probabilities of detecting the system with the initial state  $\rho_j(0)$  in the  $i$ th computational basis state  $|i\rangle$  at different times  $t$ . We model experimental and statistical errors by adding noise to the input data. The added noise is drawn from a normal distribution with mean zero and variance  $\sigma^2$ . Moreover, to be consistent with the laws of physics, we constrain the obtained noisy data such that  $0 \leq y_{i,j}(t) \leq 1$ .

To train the meta-optimizer neural network, we approximate the expectation value in the loss function (3) by the average over finitely many samples of  $f$ . Each sample corresponds to a realization of the Hamiltonian (4). In the spirit of the stochastic gradient descent (SGD) algorithm [32], we update the parameters  $\phi$  of the meta-optimizer through the estimated gradient of the loss function (3) from a single random realization of the Hamiltonian and its corresponding time series. This process is then iterated many times with different realizations of the Hamiltonian (4). The initial value of the  $\theta$  at the beginning of each epoch,  $\theta^{(0)}$ , reflects our prior knowledge about the problem. We choose  $\theta^{(0)} \sim \mathcal{N}(\theta^*, \sigma_{\text{in}}^2 I)$ , where  $\mathcal{N}(\mu, \Sigma)$  is the multivariate normal distribution with mean  $\mu$  and covariance matrix  $\Sigma$ , and  $\theta^*$  is the true value of the parameters of the Hamiltonian in that epoch. Throughout the paper, we set  $\sigma_{\text{in}} = 0.1$ , and the overall energy scale to 1. During training we consider  $\sigma = 0.001$ , 50 equally spaced times  $t \in [0, 10]$ , elements of  $\theta$  drawn identically at random from the uniform distribution  $\mathcal{U}(1, 2)$ . We train the meta-optimizer over  $10^4$  epochs, with  $T = 100$  steps in each epoch. We assess the performance of the meta-optimizer every 100 epochs and at the end choose the best performing model.

*Results.*—After initial training, we first test the meta-optimizer on the same class of Hamiltonian with the same number of qubits, but with newly sampled parameters. We compare the average values of the optimizee objective function  $f$  (given by Eq. (1)) versus the number of optimization steps (called iterations)  $k$  obtained from our

meta-optimizer and other well-known algorithms. The results, shown in Fig. 2, indicate that in most cases, the meta-optimizer outperforms derivative-free Nelder-Mead method [36], and first-order optimization methods, such as Adam [37] and SGD [32] with fine-tuned learning rate [30]. It also performs better on average than L-BFGS [38], a second-order optimization method; however, the advantage, in this case, is less clear. To better understand the performance of our LSTM optimizer, we consider the statistics of the final value of the objective function  $f$ , in addition to its average over optimization steps. We observe that while the meta-optimizer is less likely to get stuck in local minima, which leads to a lower average cost, L-BFGS attains a similar final value in cases that it does not get stuck, see histograms in Fig. 2(a). We also show how a smaller value of the objective function translates to a better estimate of the Hamiltonian parameters in Fig. 2(b).

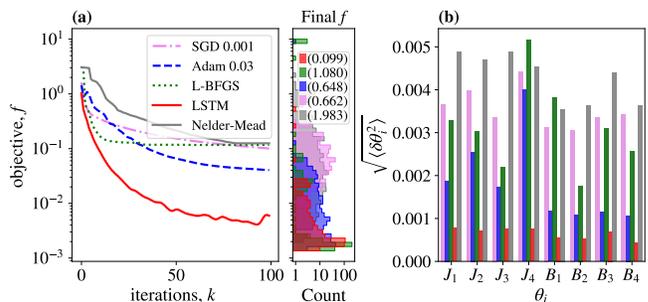


FIG. 2. *Performance of the meta-optimizer:* We test our optimizer on the same model as it was trained on, that is, on  $N = 4$  TFIM with stochastic noise  $\sigma = 0.001$ . (a) (left panel) The loss function  $f$  versus the number of iterations for different optimizers. (right panel) The histogram illustrating the loss values after the last iteration. The L-BFGS distribution with multiple instances of  $f > 10^{-2}$  corresponds to the optimizer being stuck in a local minima. Numbers in parenthesis indicate the standard deviation of the final  $f$  value. (b) The mean deviation  $\delta\theta_i^2 \equiv |\theta_i - \theta_i^*|^2$  from the true value  $\theta_i^*$  of the Hamiltonian’s parameters after the last iteration. All results in these and the following figures are averaged over 300 epochs.

The flexibility of the input dimension [30] of the meta-optimizer allows us to assess its ability to generalize to models that are different than the one used in the training. We examine how the information learned from the four-spin Ising Hamiltonian transfers to more complicated situations. We first test the generalizability of the meta-optimizer on the nearest-neighbor Ising model with more spins, see Fig. 3. We observe that the comparative advantage of LSTM persists for  $N = 6$ , however its performance deteriorates for  $N = 7$  and is comparable to Adam optimizer with fine-tuned learning rate. Moreover, we now study the generalizability of the meta-optimizer to different classes of models with the same number of qubits as the training. We consider the all-to-all Ising

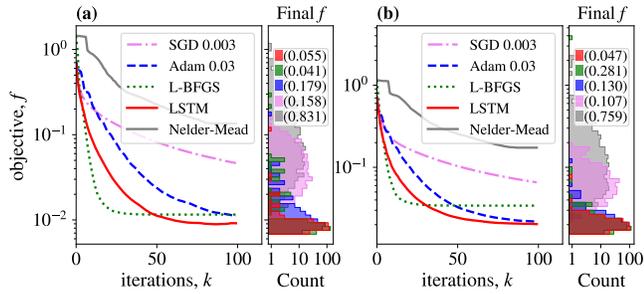


FIG. 3. *Generalizability to larger systems*: We test a model trained on  $N = 4$  qubits with the TFIM Hamiltonian (4) on larger systems with the same Hamiltonian and (a) 6 and (b) 7 qubits. The right side of each panel shows the spread of the final loss  $f$  for different optimization schemes, with the number in parenthesis indicating the corresponding standard deviations. All other parameters are the same as those in Fig. 2.

model,

$$H = \sum_{i>j}^N J_{i,j} X_i X_j + \sum_i^N B_i Z_i, \quad (5)$$

and the XY model,

$$H = \sum_i^N J_i^x X_i X_{i+1} + J_i^y Y_i Y_{i+1} + \sum_i^N Z_i. \quad (6)$$

Compared to the TFIM, in addition to different quench dynamics, these models also have more Hamiltonian parameters for a fixed system size. In Fig. 4, we see that LSTM still outperforms SGD and Adam in the mean. The performance of all of the optimizers is degraded compared with TFIM, resulting from an enhanced probability of getting stuck in local minima. Therefore, we turn to box plots (Fig. 4) to better compare the optimizer's performance in the presence of highly skewed data. We observe that boxes, which indicate the 75th and 25th percentiles are close for Adam, L-BFGS, and LSTM, however, the mean of L-BFGS is much larger than the other two because of the mentioned outliers. Note that in these plots, the whiskers show the range of the values.

Lastly, we study the robustness of the meta-optimizer against increased noise in all cases considered previously. In Fig. 5 we show results for  $\sigma = 0.003$  being three times greater than training and testing studying up to now. For all cases (except the slightly worsened performance in the all-to-all Ising model) the LSTM is still the best performing optimizer and the results are qualitatively the same as those with smaller noise in Figs. 2, 3, and 4.

*Discussion.*—The trained meta-optimizer in this work improves the classical processing resources required for calibrating experiments. Specifically, it reduces the amount of time needed to run expensive classical calculations to evaluate the cost function and its derivatives

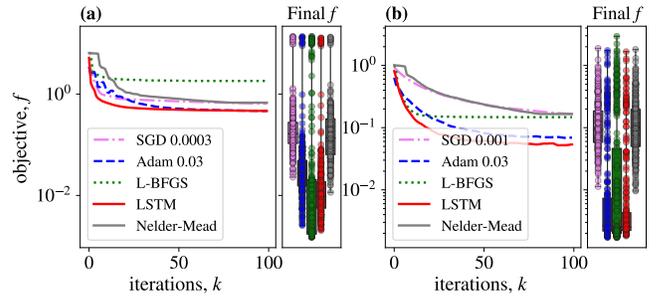


FIG. 4. *Generalizability to other models*: The model trained on 4-qubit TFIM is tested on (a) all-to-all Ising and (b) XY model. All other parameters are the same as those in Fig. 2. The panels show the statistics of the final  $f$  values. We see better results for the XY model than for all-to-all due to the similar connectivity of the XY model and the TFIM.

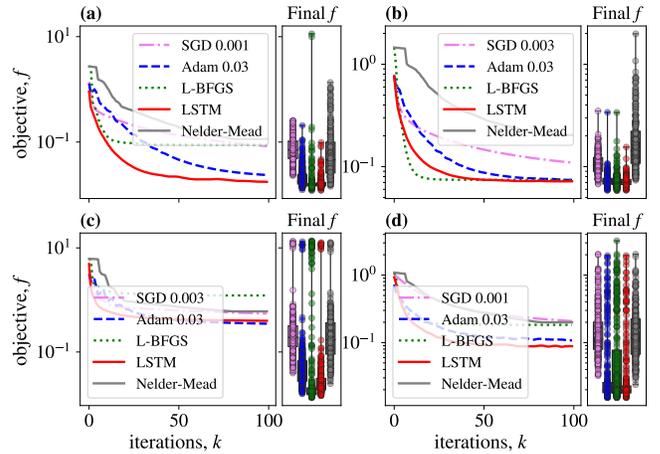


FIG. 5. *Generalizability to larger noise*: We study ten times larger stochastic noise  $\sigma = 0.003$  than for training and previous testing. (a) TFIM with  $N = 4$ , (b) TFIM for  $N = 6$ , (c) all-to-all Ising model with  $N = 4$ , and (d) XY model for  $N = 4$ .

compared to other first-order optimization methods. The optimizer is also flexible and generalizes to various models. We believe that our approach is most useful when the meta-optimizer is embedded in the daily calibration routine of an experiment, and is trained and tested on the same Hamiltonian model in the presence of considerable measurement noise.

While we have shown that our method performs well on systems larger than what it is trained on, it is ultimately limited by the classical simulability of the system of interest. In both of the training and testing stages, we used exact diagonalization to evaluate the exact derivatives of the cost function. To push the limits of the applicability of our method, it is possible to train on smaller systems where exact diagonalization and automatic differentiation are available, and use efficient numerical simulation techniques such as those based on Matrix Product

States (MPS) and Operators (MPO) [39] combined with numerical estimations of the derivatives during the test time. Another possibility is to replace the classical simulation in the test time with a well-calibrated quantum system [12] and use the output time series to estimate the gradient of the cost function numerically.

Moreover, in this work, we focused on a gradient-based technique for estimating the Hamiltonian parameters of a physical system. It is also possible to use derivative-free methods, such as Bayesian optimization for the same task [16]. Meta-learning can again be used in this context to develop task-specific optimizers [33, 40]. In this approach, it is only necessary to have gradient information during the training. Therefore, in the Hamiltonian estimation problem, it is possible to train the meta-optimizer on a small system where gradient information is readily available, and test on larger systems. The cost function, which still needs to be provided at every step in the test time, can again be evaluated using another quantum system or MPS based methods. Additionally, it might be advantageous to consider alternative meta-learning techniques to the LSTM-based approach [41, 42] in the context of Hamiltonian learning.

Finally, while we studied parameter estimation in a Hamiltonian system, it is also possible to use the techniques developed here to study open quantum systems and Lindblad learning. In this case, automatic differentiation conveniently provides a gradient of the cost function with respect to the Lindblad parameters as shown in Ref. [15], which is the main ingredient for training the meta-optimizer.

*Acknowledgments.*— We thank Patrick Becker, Norbert M. Linke, Paraj Titum, Jiehang Zhang for insightful discussion. AS and MH acknowledge support by the Physics Frontier Center at JQI, ARO-MURI and the U.S. Department of Energy, Office of Science, Quantum Systems Accelerator (QSA) program. PB acknowledges funding by DoE ASCR Accelerated Research in Quantum Computing program (award No. DE-SC0020312), U.S. Department of Energy Award No. DE-SC0019449, the DoE ASCR Quantum Testbed Pathfinder program (award No. DE-SC0019040), NSF PFCQC program, AFOSR, AFOSR MURI, and ARO MURI.

This work was completed before AS joined the University of Chicago.

---

\* These authors contributed equally.

- [1] S. Debnath, N. M. Linke, C. Figgatt, K. A. Landsman, K. Wright, and C. Monroe, “Demonstration of a small programmable quantum computer with atomic qubits,” *Nature* **536**, 63–66 (2016).
- [2] K. Wright, K. M. Beck, S. Debnath, J. M. Amini, Y. Nam, N. Grzesiak, J.-S. Chen, N. C. Pisenti, M. Chmielewski, C. Collins, et al., “Benchmarking an 11-qubit quantum computer,” *Nature communications* **10**, 1–6 (2019).
- [3] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al., “Quantum supremacy using a programmable superconducting processor,” *Nature* **574**, 505–510 (2019).
- [4] Yi Zhang and Eun Ah Kim, “Quantum Loop Topography for Machine Learning,” *Phys. Rev. Lett.* **118** (2017), 10.1103/PhysRevLett.118.216401.
- [5] Jens Eisert, Dominik Hangleiter, Nathan Walk, Ingo Roth, Damian Markham, Rhea Parekh, Ulysse Chabaud, and Elham Kashefi, “Quantum certification and benchmarking,” *Nat. Rev. Phys.* **2**, 382–390 (2020).
- [6] Jose Carrasco, Andreas Elben, Christian Kokail, Barbara Kraus, and Peter Zoller, “Theoretical and experimental perspectives of quantum verification,” arXiv preprint arXiv:2102.05927 (2021).
- [7] Marcus P. Da Silva, Olivier Landon-Cardinal, and David Poulin, “Practical characterization of quantum devices without tomography,” *Phys. Rev. Lett.* **107**, 210404 (2011).
- [8] A. Shabani, M. Mohseni, S. Lloyd, R. L. Kosut, and H. Rabitz, “Estimation of many-body quantum hamiltonians via compressive sensing,” *Phys. Rev. A* **84**, 012107 (2011).
- [9] Kenneth Rudinger and Robert Joynt, “Compressed sensing for Hamiltonian reconstruction,” *Phys. Rev. A* **92**, 052322 (2015).
- [10] Eyal Bairey, Itai Arad, and Netanel H. Lindner, “Learning a Local Hamiltonian from Local Measurements,” *Phys. Rev. Lett.* **122**, 20504 (2019).
- [11] Christopher E. Granade, Christopher Ferrie, Nathan Wiebe, and D. G. Cory, “Robust online Hamiltonian learning,” *New J. Phys.* **14**, 103013 (2012).
- [12] Nathan Wiebe, Christopher Granade, Christopher Ferrie, and D. G. Cory, “Hamiltonian learning and certification using quantum resources,” *Phys. Rev. Lett.* **112**, 190501 (2014).
- [13] Jianwei Wang, Stefano Paesani, Raffaele Santagati, Sebastian Knauer, Antonio A Gentile, Nathan Wiebe, Murrill Petruzzella, Jeremy L O’Brien, John G Rarity, Anthony Laing, et al., “Experimental quantum hamiltonian learning,” *Nature Physics* **13**, 551–555 (2017).
- [14] Ludwig E De Clercq, Robin Oswald, Christa Flühmann, Ben Keitch, Daniel Kienzler, H-Y Lo, Matteo Marinelli, David Nadlinger, Vlad Negnevitsky, and Jonathan P Home, “Estimation of a general time-dependent hamiltonian for a single qubit,” *Nature communications* **7**, 11218 (2016).
- [15] Stefan Krastanov, Sisi Zhou, Steven T. Flammia, and Liang Jiang, “Stochastic estimation of dynamical variables,” *Quantum Sci. Technol.* **4**, 035003 (2019).
- [16] Tim J. Evans, Robin Harper, and Steven T. Flammia, “Scalable Bayesian Hamiltonian learning,” arXiv:1912.07636 (2019).
- [17] Jun Zhang and Mohan Sarovar, “Quantum hamiltonian identification from measurement time traces,” *Phys. Rev. Lett.* **113**, 1–5 (2014).
- [18] Jun Zhang and Mohan Sarovar, “Identification of open quantum systems from observable time traces,” *Phys. Rev. A* **91**, 052121 (2015).
- [19] Akira Sone and Paola Cappellaro, “Hamiltonian identifiability assisted by a single-probe measurement,” *Phys. Rev. A* **95**, 022335 (2017).

- [20] Yuanlong Wang, Daoyi Dong, Akira Sone, Ian R. Petersen, Hidehiro Yonezawa, and Paola Cappellaro, “Quantum Hamiltonian Identifiability via a Similarity Transformation Approach and Beyond,” *IEEE Trans. Automat. Contr.* **65**, 4632 (2018).
- [21] Daniel Burgarth, Koji Maruyama, and Franco Nori, “Coupling strength estimation for spin chains despite restricted access,” *Phys. Rev. A* **79**, 020305 (2009).
- [22] Daniel Burgarth, Koji Maruyama, and Franco Nori, “Indirect quantum tomography of quadratic Hamiltonians,” *New J. Phys.* **13**, 0–13 (2011).
- [23] C. Di Franco, M. Paternostro, and M. S. Kim, “Hamiltonian tomography in an access-limited setting without state initialization,” *Phys. Rev. Lett.* **102**, 187203 (2009).
- [24] Sheng Tao Wang, Dong Ling Deng, and L. M. Duan, “Hamiltonian tomography for quantum many-body systems with arbitrary couplings,” *New J. Phys.* **17**, 0–9 (2015).
- [25] Xiao-Liang Qi and Daniel Ranard, “Determining a local Hamiltonian from a single eigenstate,” *Quantum* **3**, 159 (2019).
- [26] Giacomo Torlai, Guglielmo Mazzola, Juan Carrasquilla, Matthias Troyer, Roger Melko, and Giuseppe Carleo, “Neural-network quantum state tomography,” *Nature Physics* **14**, 447–450 (2018).
- [27] Juan Carrasquilla, Giacomo Torlai, Roger G Melko, and Leandro Aolita, “Reconstructing quantum states with generative models,” *Nature Machine Intelligence* **1**, 155–161 (2019).
- [28] Giacomo Torlai, Christopher J Wood, Atithi Acharya, Giuseppe Carleo, Juan Carrasquilla, and Leandro Aolita, “Quantum process tomography with unsupervised learning and tensor networks,” arXiv:2006.02424 (2020).
- [29] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas, “Learning to learn by gradient descent by gradient descent,” arXiv:1606.04474 (2016).
- [30] See Supplementary Material [url] for the technical details.
- [31] Hartmut Häffner, Christian F Roos, and Rainer Blatt, “Quantum computing with trapped ions,” *Physics reports* **469**, 155–203 (2008).
- [32] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning* (MIT Press, 2016).
- [33] Guillaume Verdon, Michael Broughton, Jarrod R. McClean, Kevin J. Sung, Ryan Babbush, Zhang Jiang, Hartmut Neven, and Masoud Mohseni, “Learning to learn with quantum neural networks via classical neural networks,” arXiv:1907.05415 (2019).
- [34] Max Wilson, Sam Stromswold, Filip Wudarski, Stuart Hadfield, Norm M. Tubman, and Eleanor Rieffel, “Optimizing quantum heuristics with meta-learning,” arXiv:1908.03185 (2019).
- [35] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio, “On the difficulty of training recurrent neural networks,” in *International conference on machine learning* (2013) pp. 1310–1318.
- [36] J. A. Nelder and R. Mead, “A Simplex Method for Function Minimization,” *Comput. J.* **7**, 308 (1965).
- [37] Diederik P Kingma, Danilo J Rezende, Shakir Mohamed, and Max Welling, “Semi-supervised Learning with Deep Generative Models,” arXiv:1406.5298 (2014).
- [38] Dong C Liu and Jorge Nocedal, “On the limited memory bfgs method for large scale optimization,” *Mathematical programming* **45**, 503–528 (1989).
- [39] Ulrich Schollwöck, “The density-matrix renormalization group in the age of matrix product states,” *Annals of physics* **326**, 96–192 (2011).
- [40] Yutian Chen, Matthew W. Hoffman, Sergio Gomez Colmenarejo, Misha Denil, Timothy P. Lillicrap, Matt Botvinick, and Nando de Freitas, “Learning to Learn without Gradient Descent by Gradient Descent,” arXiv:1611.03824 (2016).
- [41] Ke Li and Jitendra Malik, “Learning to optimize,” arXiv:1606.01885 (2016).
- [42] Ke Li and Jitendra Malik, “Learning to optimize neural nets,” arXiv:1703.00441 (2017).

## Supplemental material

In this supplement, we present technical details. For details of our LSTM construction and the effect of the size of the LSTM on the results see Sections S.I and S.V. In Sec. S.II, we discuss choosing the appropriate learning rate for the Adam and SGD optimizers. In Sec. S.III, we present a discussion on using a different loss function. In sec. S.IV, we present a discussion on the effect of the choice of initial states on the optimization.

### S.I. LSTM DETAILS

We build the meta-optimizer in our work using the coordinate-wise LSTM construction from Ref. [29]. As noted in the main text, the incremental update of variable  $\theta$  is given by  $\mathbf{g}_k$ , where  $\mathbf{g}_k$  is explicitly given by  $[\mathbf{g}_k, \mathbf{h}_{k+1}] = m(\nabla f(\theta^{(k)}), \mathbf{h}_k, \phi)$ . Here,  $m$  is modeled by an LSTM. The coordinate-wise construction essentially means that each component  $\theta_i$  has its own hidden state, but the weights of the network are shared between all the variables. Specifically, let  $m'[(\nabla^{(k)})_i, \mathbf{h}_{k,i}, \phi]$  denote an LSTM that act on a single coordinate  $\theta_i$ , where we used  $\nabla^{(k)}$  to shorten the notation of  $\nabla f(\theta^{(k)})$ . Therefore,  $\theta_i^{(k+1)} = \theta_i^{(k)} + g_{k,i}$ , where  $[g_{k,i}, \mathbf{h}_{k+1,i}] = m'((\nabla^{(k)})_i, \mathbf{h}_{k,i}, \phi)$ . We then construct the full LSTM by concatenating the LSTMs and their hidden states for each coordinate to get  $\mathbf{g}_k = [g_{k,1}, \dots, g_{k,n}]$  and  $\mathbf{h}_k = [\mathbf{h}_{k,1}, \dots, \mathbf{h}_{k,n}]$ , where  $n$  is the number of coordinates. This construction provides flexibility in input dimension, and is also invariant to the ordering of the variables [29]. To simplify the training, we also assume that at each step  $\nabla f(\theta^{(k)})$  is obtained externally, and is independent of the parameters  $\phi$  of  $m$ .

### S.II. DIFFERENT LEARNING RATES FOR ADAM AND SGD

In this section, we show results for Adam and SGD with different learning rate (see Fig. S1). Based on these results we choose the optimal learning rate  $\eta$  which is used for the comparison with L-BFGS, Nelder-Mead, and LSTM in the main text.

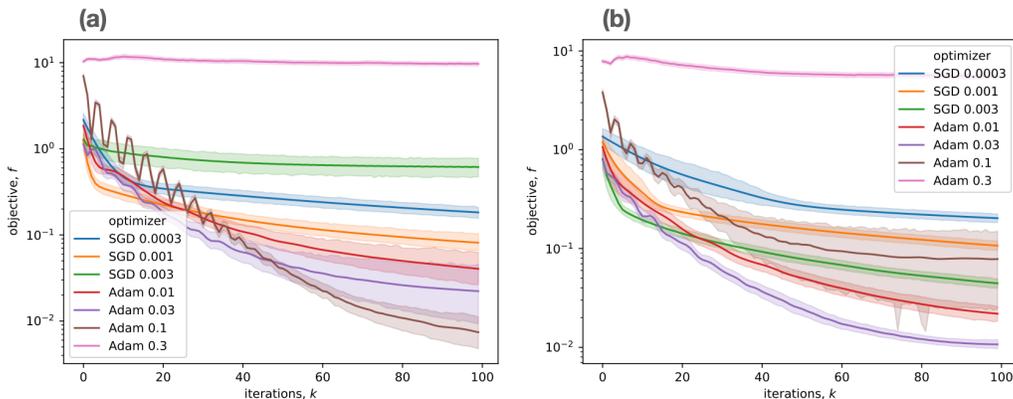


FIG. S1. Results for transverse field Ising model with stochastic noise  $\sigma = 0.001$ . The loss function  $f$  versus the number of iterations for different optimizers and different learning rates for (a)  $N = 4$  and (b)  $N = 6$ . We see that optimal learning rates are:  $\eta_{\text{Adam}} = 0.03$  and  $\eta_{\text{SGD}} = 0.001$  for  $N = 4$  and  $\eta_{\text{Adam}} = 0.03$  and  $\eta_{\text{SGD}} = 0.003$  for  $N = 6$ . The shaded region is the 95% confidence interval obtained by bootstrapping (resampling).

### S.III. KULLBACK-LEIBLER DIVERGENCE AS A COST FUNCTION

The observations we considered in this work,  $y_{ij}(t)$ , form a probability distribution over the basis states populations indexed by  $i$  for a fixed  $j$  and  $t$ . In this case, we can also use the Kullback-Leibler divergence to characterize the distance between the observed and model distributions and get

$$f_{\text{KL}}(\theta) = \sum_{j,t} D_{\text{KL}}[y_{:,j}(t) || \tilde{y}_{:,j}(t; \theta)], \quad (\text{S1})$$

where we used the notation  $y_{:,j}(t)$  to indicate the discrete population distribution over all  $i$ s at a fixed realization  $j$  and time  $t$ , and  $D_{\text{KL}}[P||Q] = \sum_i P_i \log(P_i/Q_i)$ . The test results (Fig. S2) show qualitatively similar behavior to what we saw when using the squared loss: The performance of the LSTM for  $N = 4$  is superior to other optimizers, and the advantage shrinks when a  $N = 6$  system is considered.

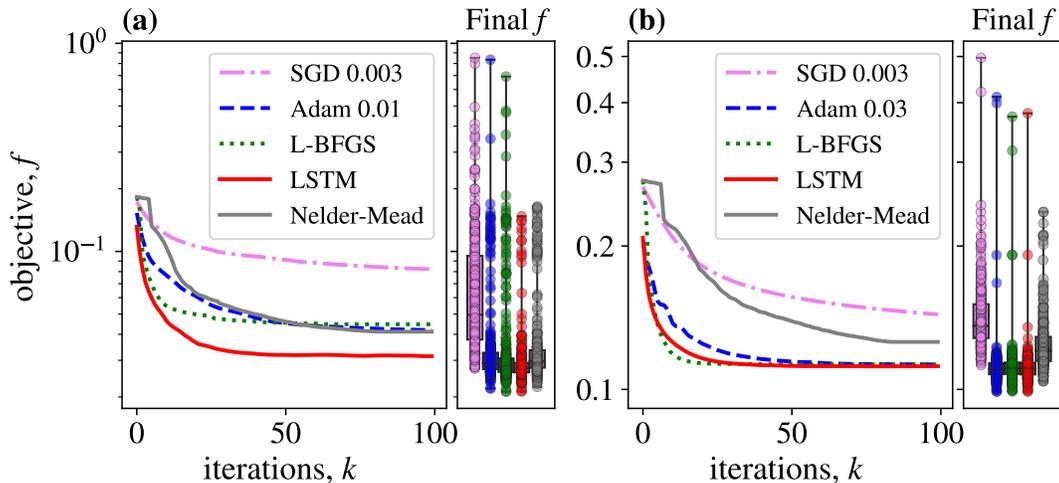


FIG. S2. Results for transverse field Ising model with stochastic noise  $\sigma = 0.001$ . The loss function proportional to the Kullback-Leibler divergence versus the number of iterations for different optimizer for  $N = 4$  and  $N = 6$ , see (a) and (b), respectively.

#### S.IV. DIFFERENT INITIAL STATES

In the main text, we study results based on two easy to prepare initial states  $\rho_j(t = 0)$  with  $j \in \{X, Z\}$ , corresponding to the state with all qubits aligned along either  $X$  or  $Z$  directions. In Fig. S3 we show the results for the training and testing based on initials state  $\rho_X(t = 0)$  and  $\rho_Z(t = 0)$ , see Fig. S3(a) and (b), respectively. We see that for testing with the same system size, LSTM outperforms the other methods in the same sense that it did in Fig. (2) in the main text.

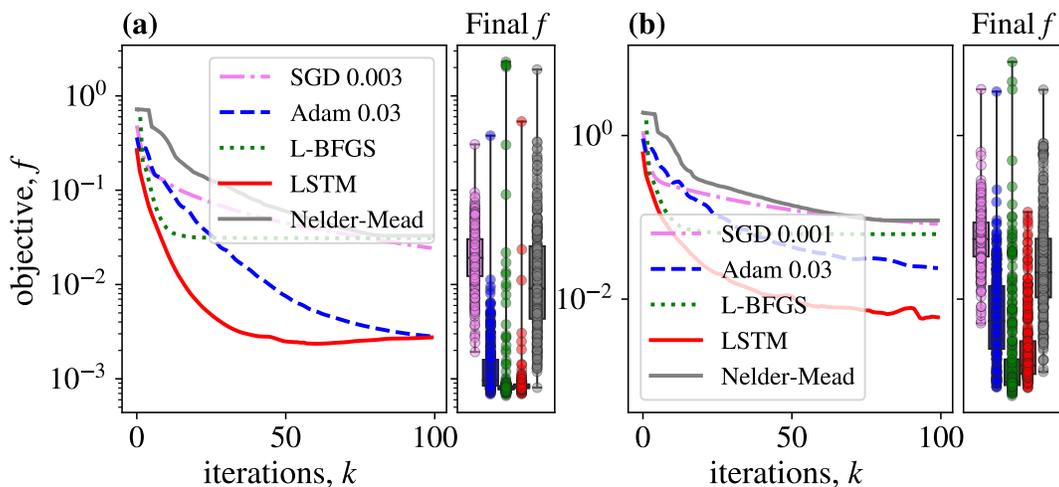


FIG. S3. Results for transverse field Ising model with stochastic noise  $\sigma = 0.001$  and  $N = 4$ . (a) in  $X$  direction, (b) in  $Z$  direction.

### S.V. DIFFERENT NUMBER OF HIDDEN NEURONS

The internal structure of the meta-optimizer can be changed. The LSTM cell is characterized by, e.g., the number of hidden neurons,  $N_h$ . All the results presented in this paper are for  $N_h = 20$ , following Ref. [29]. In Fig. S4(a) and (b), we show results for training and testing with  $N_h = 10$  and  $N_h = 30$ , respectively. We do not observe any qualitative difference between the two cases.

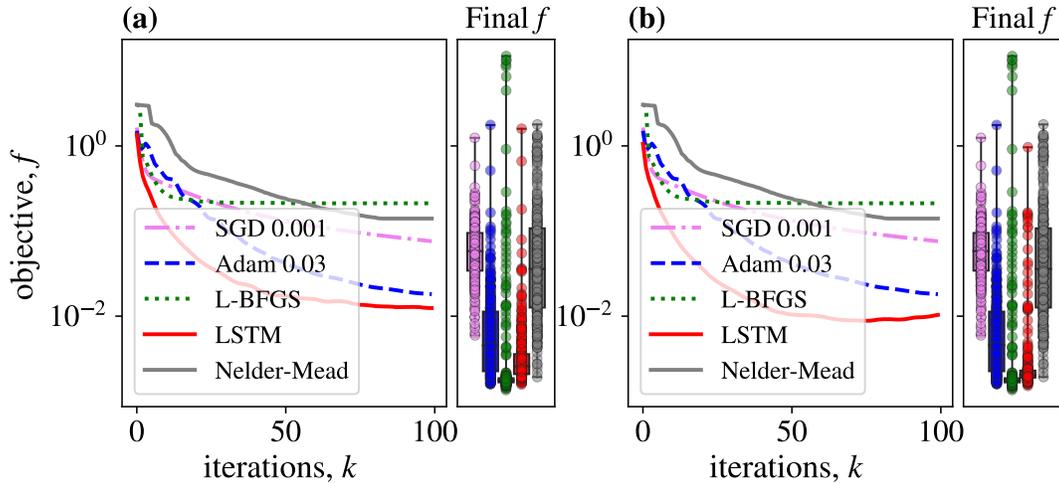


FIG. S4. Results for transverse field Ising model with stochastic noise  $\sigma = 0.001$ ,  $N = 4$ , with  $N_h = 10$  and  $N_h = 30$ , see (a) and (b), respectively.