Metalearning characteristics and dynamics of quantum systems

Lucas Schorling [1,*] Pranav Vaidhyanathan,[1,*] Jonas Schuff [2] Miguel J. Carballido [3] Dominik Zumbühl [3] Gerard Milburn [4] Florian Marquardt,[5,6] Jakob Foerster,[1] Maike Osborne [1] and Natalia Ares [1,†]

[1]*Department of Engineering Science, University of Oxford, Oxford OX1 3PJ, United Kingdom*
[2]*Department of Materials, University of Oxford, Oxford OX1 3PH, United Kingdom*
[3]*Department of Physics, University of Basel, 4056 Basel, Switzerland*
[4]*ARC Centre of Excellence for Engineered Quantum Systems, School of Mathematics and Physics, The University of Queensland, St Lucia QLD 4072, Australia*
[5]*Max Planck Institute for the Science of Light, 91058 Erlangen, Germany*
[6]*Department of Physics, Friedrich-Alexander-Universität Erlangen-Nürnberg, 91058 Erlangen, Germany*

(Received 6 June 2025; accepted 23 July 2026; published 31 August 2026)

While machine learning holds great promise for quantum technologies, most current methods focus on predicting or controlling a specific quantum system. Metalearning approaches, however, can adapt to new systems for which little data are available, by leveraging knowledge obtained from previous data associated with similar systems. In this paper, we metalearn dynamics and characteristics of closed and open two-level systems, as well as the Heisenberg model. Based on experimental data of a Loss-DiVincenzo spin qubit hosted in a Ge/Si core/shell nanowire for different gate voltage configurations, we predict qubit characteristics, i.e., g factor and Rabi frequency using metalearning. The algorithm we introduce improves upon previous state-of-the-art metalearning methods for physics-based systems by introducing techniques such as adaptive learning rates and a global optimizer for improved robustness and increased computational efficiency. We benchmark our method against other metalearning methods, a vanilla transformer, and a multilayer perceptron, and demonstrate improved performance.

DOI: 10.1103/6m8t-r5qk

I. INTRODUCTION

Recent advances in computing have enabled machine learning techniques, particularly deep learning, to make incredible progress in various sciences. These techniques have been especially useful in the field of quantum technologies [1–3]. Deep learning techniques and neural network architectures such as transformers, convolutional neural networks, and deep reinforcement learning have facilitated improved performance in quantum optimal control [4], error correction [5], simulation [6–9], and device tuning [10–15]. Specifically, recurrent neural networks have been used to reconstruct the dynamics of a quantum system [16].

One area of machine learning called metalearning, or "learning to learn," has gained significant attention in recent years in the machine learning community as a technique to account for task variation [17]. Metalearning differs from other machine learning techniques that are designed to perform task-specific predictions based on the dataset that they are trained on. This approach instead seeks to "learn" by training on *distributions* of machine learning tasks rather than individual tasks [18]. This allows for quick generalization and adaptability for various tasks. A good versatile algorithm can solve all instances of a specific task individually, but it tackles each specific task independently from the others. Metalearning, on the other hand, can exploit similar tasks seen previously to lower the quantity of new data required to tackle new tasks. This is particularly relevant for quantum technologies, since quantum device data are expensive to obtain [19]. The metalearning approach could accelerate high-throughput testing, tuning, and characterization of quantum devices, a necessity for scaling quantum technologies.

In this work, we introduce a general-purpose metalearning algorithm: Metalearning with adaptive learning rate and global optimizer (MALGO). MALGO adapts state-of-the-art metalearning approaches for physical systems [20], while introducing techniques such as adaptive learning rates and global optimizers. MALGO outperforms state-of-the-art metalearning techniques [20], a vanilla transformer [21], and a multilayer perceptron (MLP) in predicting the dynamics of three different families of quantum systems: a closed and open two-level system (TLS), and a many-body Heisenberg system. We also benchmark MALGO on data obtained from a spin-qubit device. We metalearn qubit characteristics, such as g factor and Rabi frequency f_{Rabi}, for three different device configurations. We demonstrate that MALGO performs well on structured sequence data, such as quantum dynamics, as well as unstructured set data, such as different voltages, g factor, and f_{Rabi}. The generality of the algorithm also suggests its applicability to other domains of quantum technologies.

*These authors contributed equally to this work.
†Contact author: natalia.ares@eng.ox.ac.uk

II. METALEARNING APPROACH

Among several previous metalearning techniques [22–25], the interpretable meta neural ordinary differential equation (iMODE) method has emerged as a significant leap forward in machine learning for dynamical systems [20]. Unlike alternatives such as model agnostic metalearning (MAML) [22], which adapts all network parameters, iMODE takes the core idea of rapid adaptation from fast context adaptation via metalearning [25] and applies it specifically to the domain of dynamical systems. However, it goes beyond simply applying previous algorithms to a new domain; iMODE introduces several key innovations that make it particularly effective for modeling and understanding families of related dynamical systems. The first major difference is the incorporation of neural ordinary differential equations (NODEs) into the framework. NODEs, introduced by Chen *et al.* in 2018 [26], allow for the modeling of continuous-time dynamical systems using neural networks. By combining the metalearning approach of MAML with the continuous-time modeling capabilities of NODEs, iMODE creates a powerful framework for learning and generalizing across families of dynamical systems.

Another significant contribution of iMODE is its focus on interpretability. While many machine learning models operate as "black boxes," iMODE is designed to provide insights into the underlying physics of the systems it models, via explicit "adaptation parameters." In iMODE, the model learns a shared set of parameters that capture the common form of dynamics across all instances of a system family, along with a set of system parameters that account for variations between specific instances. These system parameters form a latent space that encodes the idiosyncratic physical parameters of a given system. This approach allows iMODE not only to model the dynamics of systems accurately but also to provide interpretable insights into how those dynamics change as physical parameters vary.

The optimization procedure of iMODE can be formulated as a bilevel optimization problem. For that, the outer loop optimization focuses on learning the shared parameters that capture the common form of dynamics across system instances. The inner loop, meanwhile, adapts the model to specific system instances by optimizing the system parameters. This separation allows iMODE to learn both the general form of the dynamics and how those dynamics depend on changes in physical parameters. While adapting to new systems, iMODE only optimizes over possible physical systems. This is a significant advancement over general-purpose metalearning algorithms like MAML [22]. However, iMODE is unsuitable for handling increasingly complex physical systems due to its lack of robustness and convergence [27].

Our algorithm, MALGO, modifies iMODE by introducing an adaptive learning rate during training and a global optimizer for the system parameters during adaptation. The adaptive learning rate is motivated by lower computational cost, as well as inductive biases in line with the problem, as described in Sec. III. The adaptive learning stage comprises of three phases: random noising, updating, and freezing. The random noising shifts the focus on finding the similarities between the systems first. During updating, the systems are distinguished via common optimization. After sufficient differentiation, freezing the system parameters prevents fluctuations introduced by the outer optimization loop. For adaptation to new systems, the optimization problem is simplified. For the metalearning tasks that are the focus of this paper, it is crucial to find the global optimum because a local optimum implies a wrong estimation of system parameters. Therefore, a global optimizer is better suited than simple stochastic gradient descent, due to more reliable adaptation to new systems (see Sec. III).

III. METHODS

Let us consider a set of distinct quantum systems, denoted by $\mathcal{S}$, that exhibit variations in their Hamiltonian parameters, e.g., tunneling rates, coupling strength, and damping rates associated with a cavity. Each system $S_i \in \mathcal{S}$ encompasses several datasets of the form (X_k^i, Y_k^i), where the index k identifies individual data points for the system S_i. The data are split up into a training, adaptation, and test set. The training set $\mathcal{D}_{\text{train}}$ contains all tuples for the systems in $\mathcal{S}_{\text{train}}$. The adaptation set $\mathcal{D}_{\text{adapt}}$ contains only few tuples for each systems in $\mathcal{S}_{\text{adapt}}$. The test set $\mathcal{D}_{\text{test}}$ contains the remaining tuples of $\mathcal{S}_{\text{adapt}}$. Accessing $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{adapt}}$, metalearning aims to obtain a map $Y^j = f(X^j)$ for $S_j \in \mathcal{S}_{\text{adapt}}$. Due to the small size of $\mathcal{D}_{\text{adapt}}$, this only works if there are some underlying similarities between the different systems.

The underlying similarities of the systems, as well as their distinctiveness, can be modeled jointly via learning a parametrized function $Y^i = f_\theta(X^i; \eta_i)$. This function depends on the shared parameters θ, which capture the similarities, and the system identifiers η_i, which capture the distinctiveness of each system. θ are the parameters of a neural network with inputs X^i and η_i. This function is learned by optimizing its shared parameters θ and the system parameters η_i as shown in Fig. 1. This can be interpreted as a bilevel optimization problem with a model finding problem at the outer level (optimizing θ) and a system identification problem via parameter estimation at the inner level (optimizing η_i) with the form

$$\theta^* = \arg\min_\theta \frac{1}{n_i n_k} \sum_{S_i \in \mathcal{S}_{\text{train}}} \sum_k \left(f_\theta\left(X_k^i; \eta_i^*\right) - Y_k^i\right)^2,$$

$$\eta_i^* = \arg\min_{\eta_i} \frac{1}{n_k} \sum_k \left(f_\theta\left(X_k^i; \eta_i\right) - Y_k^i\right)^2, \quad \forall S_i \in \mathcal{S}_{\text{train}}, \quad (1)$$

where n_i and n_k are the number of systems and the number of elements per system. This optimization problem can be solved by alternating s_θ gradient-based update steps for θ followed by s_η steps for every η_i as presented in Ref. [20]. The gradients for both θ and η are obtained via backpropagation. This method does not consider higher-order derivatives, which yields efficiency speedups but estimates a biased θ gradient theoretically [20].

During adaptation, θ is frozen, and the system identification problem via parameter estimation is solved by optimizing η_j for $S_j \in \mathcal{S}_{\text{adapt}}$, namely,

$$\min_{\eta_j} \frac{1}{n_k} \sum_{k \in \mathcal{D}_{\text{adapt}}^i} \left(f_\theta\left(X_k^j; \eta_j\right) - Y_k^j\right)^2. \quad (2)$$

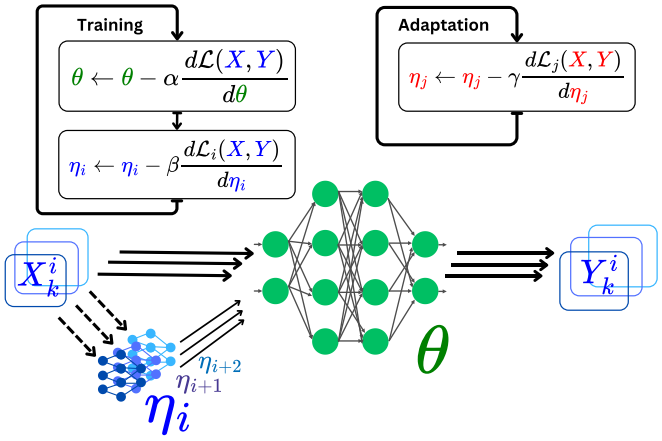

FIG. 1. Schematic representation of the metalearning algorithm during training and adaptation (with illustrative algebraic expressions). For a given system X_k^i, i denotes the system and k the index of the element. Instead of learning a single mapping $Y_k = f_\theta(X_k)$, the goal is to learn a mapping $Y_k^i = f_\theta(X_k^i; \eta_i)$, which is parametrized by η_i. Each system is assigned an initially random parameter η_i. During training, the weights of a neural network θ and all η_i are alternately updated using a gradient-based optimizer. We introduce an adaptive learning rate for this optimization. α, β, γ are the learning rates. $\mathcal{L}$ is the loss function. During adaptation, a new parameter η_j corresponding to an unseen system is optimized using a global optimizer, while θ remains fixed. When modeling a class of physical systems, θ can be interpreted as capturing the mathematical model, and η are parameters for this model assigned to a specific system instance.

With MALGO, we introduce an adaptive learning rate for η_i during training and a global optimizer for η_j during adaptation. We begin by initializing each η_i to random noise at each optimizer step. This helps focus on similarities between system instances. Optimizing the distinct system parameters to a recently initialized θ structure is computationally wasteful and introduces instabilities. Once the underlying structure of the systems is sufficiently learnt, the system identifiers can be updated via optimization to distinguish the different systems. Finally, once the systems are meaningfully distinguished, they fluctuate in a correlated manner due to changes of θ. By freezing η, we avoid the two optimization problems affecting each other.

During adaptation, estimating the parameters η_j boils down to a low-dimensional optimization problem. Therefore, we suggest using a global optimizer with better convergence properties than stochastic gradient descent and potentially less computational cost, such as restarted quasi-Newton methods [28]. Restarting is an optimization scheme that runs a local optimization algorithm for several initial values and picks the best local minimizing one.

IV. METALEARNING QUANTUM DYNAMICS

In this section, we demonstrate metalearning dynamics using MALGO. We test our algorithm for extensively studied quantum systems, namely, a closed TLS, an open TLS, and a two-spin Heisenberg model. To achieve this, we consider a set of quantum systems $\mathcal{S}_H$, whose elements undergo time evolution by an unknown family of Hamiltonians. Parameters of this Hamiltonian are varied across systems. Each system's state at time k is represented by a vector or density matrix, denoted as X_k^i. This state evolves for a fixed time interval dt to Y_k^i, the next state we want to predict.

First, for the two-level system, we consider the Hamiltonian

$$H = \Delta\sigma_x + (1-\Delta)\sigma_z, \tag{3}$$

where the parameter Δ is drawn from a uniform distribution ranging from [0,1) for each system. We train on $|\mathcal{S}_{\text{train}}| = 15$ systems and adapt on $|\mathcal{S}_{\text{adapt}}| = 35$. For each system (as well as the following ones), $\mathcal{D}_{\text{train}}$ contains (X_k^i, Y_k^i) tuples originating from five trajectories starting with random initial states and ten discrete time steps. This dataset is representatively visualized in Fig. 2(a). The adaptation set contains $|\mathcal{D}_{\text{adapt}}^j| = 3$ data points, which correspond to ∼5% of the available data for these systems that are to be metalearned.

Furthermore, we consider an open two-level system whose Lindblad master equation is given by

$$\dot{\rho} = -\frac{i}{\hbar}[\Delta\sigma_x + (1-\Delta)\sigma_z, \rho] + \gamma(\sigma_z\rho\sigma_z - \rho), \tag{4}$$

where ρ is the density matrix and γ the damping rate. Δ is drawn from an uniform distribution ranging from [0,1) and γ is drawn from an exponential distribution with a mean value of 0.2. We train on $|\mathcal{S}_{\text{train}}| = 100$ systems and adapt on $|\mathcal{S}_{\text{adapt}}| = 50$.

Finally, the two-spin Heisenberg model is given by

$$H = J_{lm} \sum_{\langle l,m \rangle} \boldsymbol{\sigma}_l \cdot \boldsymbol{\sigma}_m + \sum_{l=1}^{n_q} c_l \sigma_{l,z}, \tag{5}$$

where J_{lm} and c_l are all drawn from the uniform distribution ranging from [0,1), $\langle l, m \rangle$ indexes nearest neighbors, and $n_q = 2$. We train on $|\mathcal{S}_{\text{train}}| = 300$ systems and adapt on $|\mathcal{S}_{\text{adapt}}| = 50$. Results for larger spin chains can be found in Appendix E.

For all families of systems, training is run for 250 epochs with alternating one update step for θ and ten steps for each η_i. The quantum states are converted to real vectors to calculate an elementwise mean squared error as in Eq. (1). Figure 2(b) displays how η_i changes due to the adaptive learning rate and optimization during training for the closed TLS for the three different stages of noising, updating, and freezing. As soon as the updating starts after the initial noising, all η_i split up quickly. After some epochs, the systems are successfully distinguished, and their relative values do not change. Correlated fluctuations are observed due to small changes in θ until η_i are frozen.

For the case of simulated data, the true parameters for the different systems are known and can be compared to the assigned η_i [see Fig. 2(c)]. For low values of Δ, the algorithm extrapolates successfully from only having seen systems with higher Δ. While we expect to see a bijective relationship in case of sucessful training, there is no reason to expect a direct linear mapping or even 1:1 correspondence, especially for more than one physical parameters.

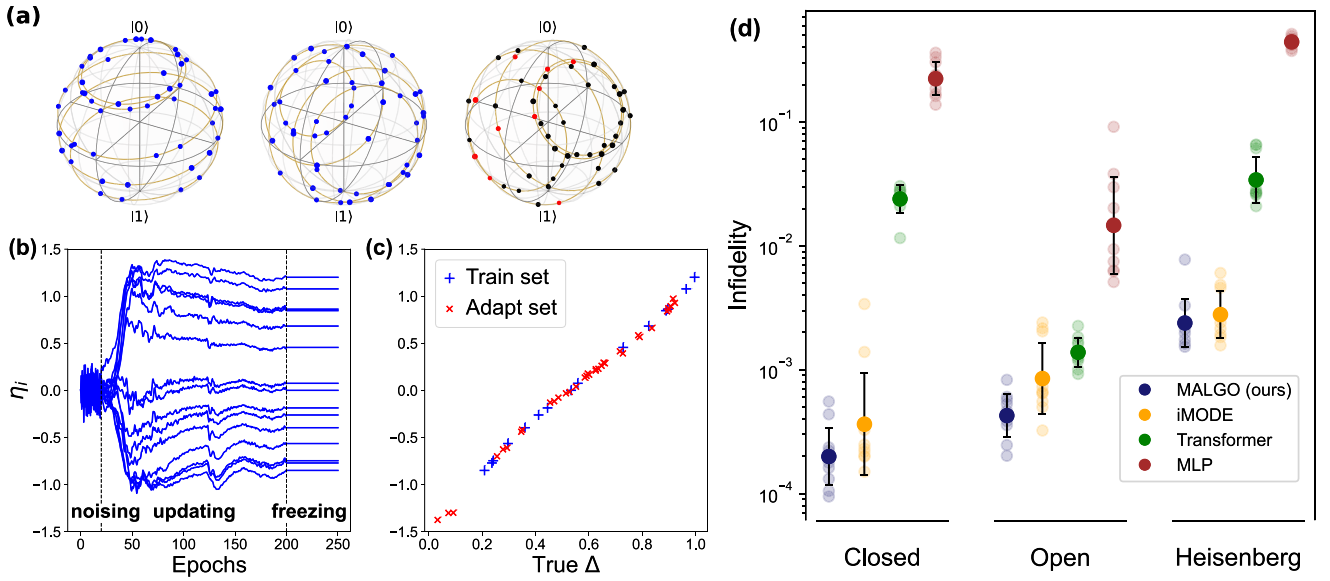

FIG. 2. (a) Different trajectories of two-level systems evolved by the Hamiltonian from Eq. (3) are represented on a Bloch sphere for $\Delta = 0.2, 0.5, 0.8$ in left-to-right order. A representative part of the datasets are visualized with two training systems (in blue) and a single adaptation system consisting of the adaptation set (in red) and test set (in black). (b) The evolution of different η_i during training is displayed. During the first 20 epochs, η_i is set to random values. From epoch 21 to 200, through the optimization, all η_i split up and reach stable relative differences. From epoch 201 on, all η_i are frozen and only the weights of the neural network are still updated. (c) The assigned η_i is compared with the true parameter Δ in the Hamiltonian. We show both the η_i from the training set, as well as the assigned η_i for the adapted systems. The monotone function suggests that the metalearning algorithm learned an interpretable representation for all systems. The algorithm can both interpolate and extrapolate to new systems. (d) We benchmark the algorithm against three other methods for the considered quantum systems and report the infidelity of predicted vs actual quantum state on the test set. The faded points correspond to the average infidelity over all the test systems and data points for a single run.

We benchmark MALGO for the closed and open TLS, as well as the Heisenberg model against iMODE, a vanilla transformer with known metalearning abilities [29] and an MLP, with no metalearning capabilities. Every algorithm is run 10 times for every set of systems. In Fig. 2(d), we report average infidelity for every run and their mean and standard deviation on a logarithmic scale.

For the transformer, we feed data tuples (X_k^i, Y_k^i) for system i into the encoder, feed X_l^i into the decoder, and try to predict Y_l^i. This process is used for both training and adaptation. The number of data points remains consistent across training and adaptation for the encoder. For the MLP, the training set is discarded and trained from scratch for every system on the small adaptation set and then tested on the test set.

For all sets of systems, our method outperforms iMODE, the vanilla transformer, and the MLP. For the closed and open TLS cases, the standard deviation of our method is lower than the iMODE version that indicates a more stable process. The vanilla transformer fails to exhibit optimal performance under the data constraints. Nevertheless, the transformer architecture is yet to be experimentally applicable to real-time control of quantum systems due to its large size and context requirements [4]. The compelling performance of MALGO as compared to MLP confirms the benefits of metalearning compared to learning from scratch. MALGO performs the best on the closed TLS Hamiltonian with one free parameter. Larger training sets that better cover the parameter space of the Hamiltonians might further improve the performance for the open and Heisenberg problem.

V. METALEARNING SPIN QUBIT CHARACTERISTICS

Qubits encoded in gated semiconductor devices are tunable via gate voltages [30]. In devices with strong spin-orbit coupling and strong two-dimensional confinement, such as Ge/Si core/shell nanowires, the qubit Larmor frequency, expressed through the g factor, and its Rabi frequency f_{Rabi} strongly depend on these gate voltages due to their influence on the local electric field and potential landscape [31,32].

Deep learning algorithms without metalearning ability can predict qubit characteristics only for trained gate voltage configurations with fixed plunger gate voltages and therefore the same discrete charge configurations. However, MALGO equipped with metalearning allows us to extrapolate to entirely *different* gate voltage configurations, and learn the correlation between experimentally controllable parameters and quantum system characteristics. This proves that our underlying algorithm effectively operates on unstructured set data, as mentioned in Sec. I.

We utilize data from a spin qubit for three different voltage configurations [see Fig. 3(a)]. This qubit is a Loss-DiVicenzo hole spin qubit hosted in a double quantum dot formed in a Ge/Si core/shell nanowire [31]. The different gate voltage configurations correspond to the charge configurations of the double quantum dot, making the systems different but with underlying similarities.

For training purposes, gate voltages, g factor, and f_{Rabi} were normalized, to zero mean and standard deviation of one, for each gate voltage configuration. More details about the

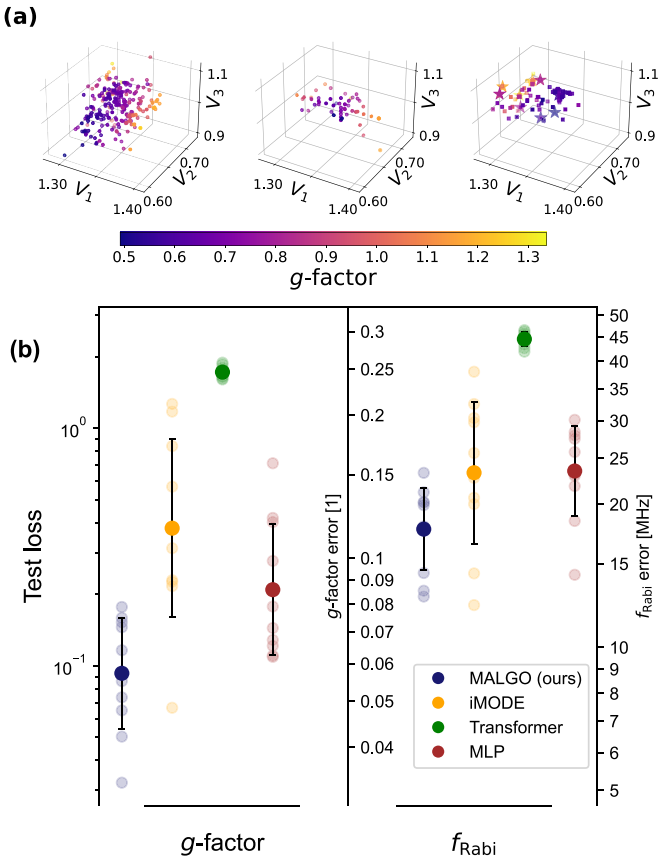

FIG. 3. (a) For three gate voltage configurations, the mapping from barrier gate voltages V_1, V_2, and V_3 to g factor is depicted via color encoding. The shape of the points indicates the different sets, where circle, star, and square correspond to training, adaptation, and test set, respectively. (b) We benchmark MALGO against three other methods for both the g factor and f_{Rabi} mapping. Single runs, as well as their average and standard deviation, are shown. The relationship between test loss and g-factor error or f_{Rabi} error is also visualized.

physical device, measurement methods for the g factor and f_{Rabi}, and the dataset size can be found in Appendix B.

The hyperparameters are the same as in Sec. IV unless stated. η_i is chosen to be seven dimensional, and 7% of the data points in the last transition belong to $\mathcal{D}_{\text{adapt}}$. Conceptually, ideal performance is achieved when the dimensionality of η_i is at least as large as the unknown latent dimensionality of the physical problem [20]. An ablation study examining adaptation set of different sizes can be found in Appendix D.

We evaluate our method in comparison to the three other methods employed previously. We present the average losses for each iteration in Fig. 3(b). For both the g factor and f_{Rabi}, our method demonstrates the best performance. We notice better performance of all algorithms for the g factor dataset due to less noisy data as compared to that of f_{Rabi}. MALGO exceeds the performance on these datasets against several state-of-the-art methods.

VI. CONCLUSION AND OUTLOOK

In this paper, we introduced MALGO, a metalearning algorithm with an adaptive learning rate and global optimizer, and benchmarked it on predicting quantum dynamics and qubit characteristics. This algorithm outperforms the traditional iMODE, vanilla transformer, as well as a simple MLP at predicting the dynamics of new quantum systems from little data. Furthermore, we applied our algorithm to experimental set data, namely, gate voltages and their corresponding effect on g factor and f_{Rabi}. The algorithm successfully metalearned this mapping to new gate voltage configurations. Metalearning is the natural choice to tackle the inherent variability of many tasks across quantum technologies.

ACKNOWLEDGMENTS

N.A. acknowledges support from the European Research Council (Grant Agreement No. 948932) and the Royal Society (URF-R1-191150). P.V. was supported by the United States Army Research Office under Award No. W911NF-21-S-0009-2. L.S. was supported by DTP Grant No. EP/W524311/1. This work was also supported by the Swiss NSF through the NCCR SPIN Grant No. 225153 and the EU H2020 European Microkelvin Platform EMP Grant No. 824109. Views and opinions expressed are, however, those of the authors only and do not necessarily reflect those of the European Union, Research Executive Agency, or UK Research and Innovation. Neither the European Union nor UK Research and Innovation can be held responsible for them.

DATA AVAILABILITY

The data associated with model architecture, training, data generation, and experimental datasets are publicly available [33].

APPENDIX A: HYPERPARAMETERS DYNAMICS

The network architecture is DenseNet-like [34] and the same as in Ref. [20] with seven hidden layers and a first layer size of 25. The core idea is the concatenation of the outputs of all previous layers into a long vector as input into the next layer. The transformer has a model dimension of 64, with four layers in the encoder and decoder and a feedforward dimension of 128. The network architecture for the MLP is a fully connected multilayer perceptron with seven hidden layers and a layer size of 50. Our method, iMODE and the MLP architecture have around 15k parameters, and the transformer is above 300k.

During training of MALGO, we use the adaptive moment estimator (ADAM) optimizer for both θ and η with learning rates of 0.01 and 0.003. For adaptation, we use the global optimizer limited-memory Broyden-Fletcher-Goldfarb-Shanno [35] for 10 epochs with five different starts and with line search using the Wolfe condition. For iMODE, transformer, and MLP, the ADAM optimizer is used during training [36]. For the iMODE method, stochastic gradient descent (SGD) for adaptation with 1000 steps and a learning rate of 0.1 is used. The transformer optimizer has a learning rate of 0.001 and a dropout of 0.1. The learning rate for the MLP optimizer is 0.01.

The batch size is 500 for the closed TLS, 1000 for the open TLS, and 3000 for the Heisenberg model for all the algorithms benchmarked. The dimension of η corresponds to

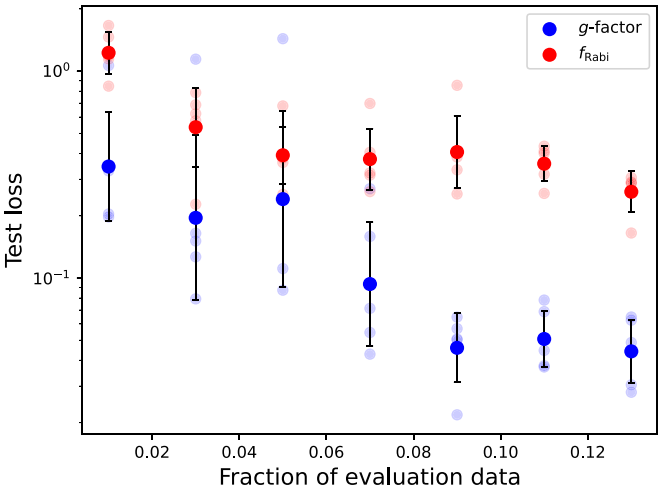

FIG. 4. Test loss for varying sizes of the adaptation set.

the number of parameters in the Hamiltonian, namely, 1, 2, and 3 for closed, open, and Heisenberg systems, respectively. All experiments in the main text were run on a MacBook Pro with an Apple M3 Pro chip and 18 GB of RAM.

APPENDIX B: EXPERIMENTAL DETAILS

The quantum device is a Ge/Si core/shell nanowire spanned across nine electrically tunable voltage gates. A hole double quantum dot can be formed by tuning the five left gates. The experimental setup is the same as described in detail in Ref. [31]. The five gates are split into three barriers and two plunger gates. Barrier gates mainly control the interaction between the source and drain with the double quantum dot as well as the interaction of the dots. The two plunger gates mainly control the chemical potential of the quantum dots.

The g factors are measured via electric dipole spin resonance scans. Rabi frequencies are extracted via parameter fitting for current traces. The three voltages in the dataset correspond to the three barrier gates. The plunger gates are chosen such that for each gate voltage configuration, the quantum double dot has a different charge occupation. The specific values for the plunger gates are chosen to maximize the quality of the measurement. Details about the measurement pipeline can be found in Ref. [19].

The distribution of points in barrier voltage space and their quantities for each gate voltage configuration are heterogeneous, due to gathering data across several experiments. Sampling techniques include quasirandom sequences, Bayesian optimization, and evaluations along specific lines.

The data are noisy and both characteristics were sometimes evaluated multiple times for the same voltages with up to 20% measured differences, in particular for f_{Rabi}. The dataset of g factors contains 260, 84, and 125 data points, and the dataset for f_{Rabi} contains 215, 33, and 97 data points for each gate voltage configuration, respectively.

APPENDIX C: HYPERPARAMETERS CHARACTERISTICS

Network architecture is as in Appendix A with six hidden layers and a first layer size of 15. The transformer architecture remains unchanged. The network architecture for the MLP is a fully connected multilayer perceptron with six hidden layers and a layer size of 28. During adaptation, our optimizer uses 20 restarts with 2 epochs. The iMODE optimizer uses SGD with 1000 epochs. A batch size of 200 is used throughout.

TABLE I. Hyperparameters for the two-qubit base case and their per-qubit scaling rule ($\Delta n = n_{\mathrm{qubits}} - 2$).

Hyperparameter	Initial value ($n=2$)	Scaling
Hidden layer width H	25	$\times 2^{\Delta n}$
Training epochs	250	$\times(\sqrt{2})^{\Delta n}$
Freeze epochs	200	$\times(\sqrt{2})^{\Delta n}$
Batch size	500	$\times 2^{\Delta n}$
Learning rate η	0.01	$\times(1/\sqrt{2})^{\Delta n}$
Global opt. restarts	5	Fixed
Total systems	50	Fixed
Training systems	25	Fixed
Initial states per system	15	$\times 2^{\Delta n}$

APPENDIX D: ABLATION STUDY ADAPTATION SET SIZE

As seen in Fig. 4, we report the test loss for varying sizes of the adaptation set for both g factor and f_{Rabi}. The mean and standard deviation, computed on a logarithmic scale over five runs, are shown, with individual results displayed in a faded manner. Overall, the test loss decreases as $|\mathcal{S}_{\mathrm{adapt}}|$ increases.

APPENDIX E: ABLATION STUDY SIZE OF HEISENBERG MODEL AND SCALING BEHAVIOUR

We conduct a scaling ablation for longer Heisenberg chains. We use the Hamiltonian from Eq. (5) with index-independent parameters, $J_{lm} = J$ and $c_l = c$. Table I reports the hyperparameters for two qubits as well as the corresponding scaling factor. Based on these heuristics, the number of epochs scales with $\sqrt{2}$ for every extra qubit, and the number of gradient steps per epoch (on the outer loop level) stay constant as both dataset size and batch size double. Learning rate and number of update steps in the inner loop remain constant.

For those sets of hyperparameters, Table II reports the final infidelity, the number of trainable parameters, overall training time, training time per gradient step (of the outer loop), and compute time of adaptation. All experiments were executed on an NVIDIA RTX 4000 Ada Generation GPU.

TABLE II. Scaling of final infidelity, network size, training time, time per gradient step (of the outer loop), and adaptation time with the number of qubits on the Heisenberg model.

	Two qubits	Three qubits	Four qubits
Final infidelity	2.56×10^{-3}	5.07×10^{-3}	3.04×10^{-2}
Trainable parameters	14,958	59 366	236 532
Training time (s)	128.6	191.2	358.5
Time per GS (ms)	73.5	77.6	103.1
Adaptation time (s)	3.7	3.6	15.0

[1] Y. Alexeev, M. H. Farag, T. L. Patti, M. E. Wolf, N. Ares, A. Aspuru-Guzik, S. C. Benjamin, Z. Cai, Z. Chandani, F. Fedele, *et al.*, Artificial intelligence for quantum computing, Nat. Commun. **16**, 10829 (2025).

[2] M. Krenn, J. Landgraf, T. Foesel, and F. Marquardt, Artificial intelligence and machine learning for quantum technologies, Phys. Rev. A **107**, 010101 (2023).

[3] V. Gebhart, R. Santagati, A. Gentile, E. Gauger, D. Craig, N. Ares, L. Banchi, F. Marquardt, L. Pezze, and C. Bonato, Learning quantum systems, Nat. Rev. Phys. **5**, 141 (2023).

[4] P. Vaidhyanathan, F. Marquardt, M. T. Mitchison, and N. Ares, Quantum feedback control with a transformer neural network architecture, Phys. Rev. Res. **8**, L012043 (2026).

[5] F. Marquardt, Machine learning and quantum devices, SciPost Phys. Lect. Notes, 29 (2021).

[6] D. L. Craig, H. Moon, F. Fedele, D. T. Lennon, B. van Straaten, F. Vigneau, L. C. Camenzind, D. M. Zumbühl, G. A. D. Briggs, M. A. Osborne, D. Sejdinovic, and N. Ares, Bridging the reality gap in quantum devices with physics-aware machine learning, Phys. Rev. X **14**, 011001 (2024).

[7] D. L. Craig, N. Ares, and E. M. Gauger, Differentiable master equation solver for quantum device characterization, Phys. Rev. Res. **6**, 043175 (2024).

[8] B. Flynn, A. A. Gentile, N. Wiebe, R. Santagati, and A. Laing, Quantum model learning agent: Characterisation of quantum systems through machine learning, New J. Phys. **24**, 053034 (2022).

[9] B. van Straaten, J. Hickie, L. Schorling, J. Schuff, F. Fedele, and N. Ares, QArray: A GPU-accelerated constant capacitance model simulator for large quantum dot arrays, SciPost Phys. Codebases, 35 (2024).

[10] N. Ares, Machine learning as an enabler of qubit scalability, Nat. Rev. Mater. **6**, 870 (2021).

[11] J. Schuff, M. J. Carballido, M. Kotzagiannidis, J. C. Calvo, M. Caselli, J. Rawling, D. L. Craig, B. van Straaten, B. Severin, F. Fedele, S. Svab, P. C. Kwon, R. S. Eggli, T. Patlatiuk, N. Korda, D. Zumbühl, and N. Ares, Fully autonomous tuning of a spin qubit, Nat. Electron. **9**, 304 (2026).

[12] V. Nguyen, S. B. Orbell, D. T. Lennon, H. Moon, F. Vigneau, L. C. Camenzind, L. Yu, D. M. Zumbühl, G. A. D. Briggs, M. A. Osborne, D. Sejdinovic, and N. Ares, Deep reinforcement learning for efficient measurement of quantum devices, npj Quantum Inf. **7**, 100 (2021).

[13] B. van Straaten, F. Fedele, F. Vigneau, J. Hickie, D. Jirovec, A. Ballabio, D. Chrastina, G. Isella, G. Katsaros, and N. Ares, All RF-based tuning algorithm for quantum devices using machine learning, Phys. Rev. Appl. **24**, 054030 (2025).

[14] J. Schuff, D. T. Lennon, S. Geyer, D. L. Craig, F. Fedele, F. Vigneau, L. C. Camenzind, A. V. Kuhlmann, G. A. D. Briggs, D. M. Zumbühl, D. Sejdinovic, and N. Ares, Identifying Pauli spin blockade using deep learning, Quantum **7**, 1077 (2023).

[15] J. P. Zwolak and J. M. Taylor, Colloquium: Advances in automation of quantum dot devices control, Rev. Mod. Phys. **95**, 011006 (2023).

[16] E. Flurin, L. S. Martin, S. Hacohen-Gourgy, and I. Siddiqi, Using a recurrent neural network to reconstruct quantum dynamics of a superconducting qubit from physical observations, Phys. Rev. X **10**, 011006 (2020).

[17] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, Meta-learning in neural networks: A survey, IEEE Trans. Pattern Anal. Mach. Intell. **44**, 5149 (2022).

[18] S. Thrun and L. Pratt, Learning to learn: Introduction and overview, in *Learning to Learn*, edited by S. Thrun and L. Pratt (Springer US, Boston, MA, 1998), pp. 3–17.

[19] J. Schuff, Characterisation and fully autonomous tuning of spin qubits with machine learning, Ph.D. thesis, University of Oxford, 2024.

[20] Q. Li, T. Wang, V. Roychowdhury, and M. K. Jawed, Metalearning generalizable dynamics from trajectories, Phys. Rev. Lett. **131**, 067301 (2023).

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, Attention is all you need, in *Advances in Neural Information Processing Systems*, Vol. 30, edited by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Curran Associates, Inc., 2017).

[22] C. Finn, P. Abbeel, and S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in *International Conference on Machine Learning* (PMLR, 2017), pp. 1126–1135.

[23] A. Nichol, J. Achiam, and J. Schulman, On first-order meta-learning algorithms, arXiv:1803.02999.

[24] A. Rajeswaran, C. Finn, S. M. Kakade, and S. Levine, Meta-learning with implicit gradients, in *Advances in Neural Information Processing Systems*, Vol. 32 (NeurIPS, 2019).

[25] L. M. Zintgraf, K. Shiarlis, V. Kurin, K. Hofmann, and S. Whiteson, CAML: Fast context adaptation via meta-learning, arXiv:1810.03642.

[26] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, Neural ordinary differential equations, in *Advances in Neural Information Processing Systems* (Curran Associates, Inc., 2018), Vol. 31.

[27] P. Vaidhyanathan, A. Papatheodorou, M. T. Mitchison, N. Ares, and I. Havoutis, MetaSym: A symplectic meta-learning framework for physical intelligence, arXiv:2502.16667.

[28] J. M. Martınez, Practical quasi-Newton methods for solving nonlinear systems, J. Comput. Appl. Math. **124**, 97 (2000).

[29] L. C. Melo, Transformers are meta-reinforcement learners, in *International Conference on Machine Learning* (PMLR, 2022), pp. 15340–15359.

[30] G. Burkard, T. D. Ladd, A. Pan, J. M. Nichol, and J. R. Petta, Semiconductor spin qubits, Rev. Mod. Phys. **95**, 025003 (2023).

[31] M. J. Carballido, S. Svab, R. S. Eggli, T. Patlatiuk, P. C. Kwon, J. Schuff, R. M. Kaiser, L. C. Camenzind, A. Li, N. Ares, E. P. A. M. Bakkers, S. Bosco, J. C. Egues, D. Loss, and D. M. Zumbühl, Compromise-free scaling of qubit speed and coherence, Nat. Commun. **16**, 7616 (2025).

[32] F. N. Froning, L. C. Camenzind, O. A. van der Molen, A. Li, E. P. Bakkers, D. M. Zumbühl, and F. R. Braakman, Ultrafast hole spin qubit with gate-tunable spin–orbit switch functionality, Nat. Nanotechnol. **16**, 308 (2021).

[33] L. Schorling, P. Vaidhyanathan, J. Schuff, M. J. Carballido, D. Zumbühl, G. Milburn, F. Marquardt, J. Foerster, M. Osborne, and N. Ares, Code repository, GitHub, 2026, https://github.com/blueluckyluke/malgo.

[34] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, Densely connected convolutional networks, in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, Washington DC, 2017), pp. 2261–2269.

[35] D. C. Liu and J. Nocedal, On the limited memory BFGS method for large scale optimization, Math. Program. **45**, 503 (1989).

[36] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980.