MACHINE LEARNING Science and Technology

PAPER

OPEN ACCESS

RECEIVED
4 December 2020

REVISED
22 January 2021

ACCEPTED FOR PUBLICATION
12 February 2021

PUBLISHED
13 May 2021

Machine learning pipeline for quantum state estimation with incomplete measurements

Onur Danaci[1], Sanjaya Lohani[1], Brian T Kirby[1,2] and Ryan T Glasser[1]

[1] Tulane University, New Orleans, LA 70118, United States of America
[2] United States Army Research Laboratory, Adelphi, MD 20783, United States of America

E-mail: brian.t.kirby4.civ@mail.mil and rglasser@tulane.edu

Keywords: quantum state estimation, quantum state tomography, machine learning, convolutional neural networks, multiple imputation, extreme gradient boosting, stacked generalization

Abstract

Two-qubit systems typically employ 36 projective measurements for high-fidelity tomographic estimation. The overcomplete nature of the 36 measurements suggests possible robustness of the estimation procedure to missing measurements. In this paper, we explore the resilience of machine-learning-based quantum state estimation techniques to missing measurements by creating a pipeline of stacked machine learning models for imputation, denoising, and state estimation. When applied to simulated noiseless and noisy projective measurement data for both pure and mixed states, we demonstrate quantum state estimation from partial measurement results that outperforms previously developed machine-learning-based methods in reconstruction fidelity and several conventional methods in terms of resource scaling. Notably, our developed model does not require training a separate model for each missing measurement, making it potentially applicable to quantum state estimation of large quantum systems where preprocessing is computationally infeasible due to the exponential scaling of quantum system dimension.

1. Introduction

The intersection of classical machine learning (ML) and quantum information science (QIS) has recently become an area of intense investigation [1]. Application areas are diverse and include, for example, the representation and classification of many-body quantum states [2], the verification of quantum devices [3], quantum error correction [4], quantum control [5], and quantum state tomography (QST) [6, 7]. Early results indicate that ML approaches to processing classical information associated with executing QIS protocols may have advantages compared with standard methods such as improved resource scaling [8] and resilience to noise [6].

Estimating an unknown quantum state using QST requires repeated joint measurements on an ensemble of identically prepared quantum systems [7, 9]. The computational resources required for state estimation alone, after the experimental aspects of QST have been performed, scale poorly even in situations where specific noise models are assumed [10–12]. Recently, various ML approaches for quantum state estimation have been proposed [6, 13–22] with some techniques indicating a scaling of $O(d^3)$ [8].

In principle, two-qubit state tomography is possible from the statistics of only 16 measurements [23, 24]. However, motivated by results showing that the use of mutually unbiased bases for state estimation minimizes statistical error, two-qubit state tomography is typically performed using 36 different projective measurement settings [23, 25]. Better understanding the reliance of state estimation fidelity on the number of input measurements is an essential question for QST on large quantum systems. Enabling an experimentalist to determine how costly avoiding specific measurement settings would be for a given experiment is of significant value.

In general, incomplete QST will not specify a unique state, requiring an additional constraint to decide between physically valid estimations. One of the most popular approaches is the maximum entropy principle, which finds the state consistent with the measured data with the largest von Neumann entropy

[26]. Alternatively, variational quantum tomography attempts to find a physically valid state that minimizes the expectation value of the missing projectors [27, 28]. Similarly, methods that jointly maximize the likelihood and the entropy have been explored [29, 30]. Additionally, under the assumption of high purity, compressed sensing techniques become practical [31, 32]. Following an entirely different paradigm in our previous work, we showed that training a neural network on simulated tomography data, in principle, results in estimators that still yield high-fidelity quantum state estimation when some measurements are missing [6]. Nevertheless, this problem required training a different model for each combination of missing data. The necessity of $\binom{36}{k}$ for k missing measurements makes it computationally impractical when moving to large dimensional systems. Here we aim to solve this issue with a limited stack of models and data imputation.

In this paper, we diversify and significantly expand our machine learning models and deploy them within a pipeline for the task of fast and robust quantum state estimation from the partial tomography measurements of high-dimensional quantum systems. Our approach differs from our previous work, where we deployed machine learning techniques for tomographic estimation of two-qubit states [6], mainly through the decoupling of various functionalities gathered in one single model into different models. First, we trained a one-dimensional convolutional neural network (Conv1D-Regressor), a two-dimensional CNN (Conv2D-Regressor), and an extreme gradient boosting regression model (XGB-Regressor) for pure and mixed quantum states using noiseless, simulated measurement data. We trained three regression models for pure and mixed states, respectively, totaling six regression models for the task of quantum state estimation from noiseless projective measurements. Each of these models is sensitive to different types of projective measurements, i.e. predictors or features. Second, we trained a Conv1D based denoising autoencoder (Conv1D-Denoise), which takes noisy data and outputs noiseless data, using noisy and noiseless simulated measurement data. Third, we trained a Conv1D based classifier to tell whether a sample of projective measurements (36 measurements or features) is noisy or not (Conv1D-isnoise). Fourth, we trained another Conv1D based classifier to tell whether a sample is a pure or mixed state (Conv1D-ispure). Last, we trained a multivariate imputation by a chained equation model based on extreme gradient boosting estimators (XGB-MICE) to predict missing measurements using numerically synthesized experimental data.

A schematic of our developed pipeline is shown in figure 1. After training all the models in the pipeline, a sample of a noisy and incomplete set of projective measurements is first complemented by the XGB-MICE model during the testing stage. Next, the complemented sample moves to the Conv1D-isnoise classifier; if it is deemed to be noisy, it moves to Conv1D-Denoise to be denoised; else, it moves to Conv1D-ispure. If the Conv1D-ispure model deems the sample a pure state, then the sample proceeds to three fast-QST models trained on pure states (Conv1D, Conv2D, XGB regressors), else it proceeds to their equivalents for mixed states. At the end of the pipeline, three output reconstructed states are obtained from Conv1D, Conv2D, XGB regressors.

Our pipeline is a promising technique for reducing the computational resources required for quantum state estimation. Having trained on synthesized data, we reconstruct states that outperform our previously developed machine-learning-based approach in reconstruction fidelity and several conventional methods in terms of resource scaling. Further, our unique method allows for significant resilience to missing and noisy data without the need to train $\binom{36}{k}$ different models for each k missing measurements.

2. Methods

2.1. Simulating measurement results

We define an arbitrary set of orthogonal basis states as

$$|U_+\rangle = \begin{bmatrix}1\\0\end{bmatrix}, \quad \text{and} \quad |U_-\rangle = \begin{bmatrix}0\\1\end{bmatrix}. \tag{1}$$

The orthogonal vectors $|U_+\rangle$ and $|U_-\rangle$ can be supplemented with $|V_\pm\rangle = (|U_+\rangle \pm |U_-\rangle)/\sqrt{2}$ and $|W_\pm\rangle = (|U_+\rangle \pm i|U_-\rangle)/\sqrt{2}$ to form a mutually unbiased bases (MUB). Further, we notate the tensor product $|U_\pm\rangle \otimes |V_\pm\rangle$ as $|U_\pm V_\pm\rangle$, with the other bases combinations treated similarly. These definitions are general and independent of physical implementation, making them equally applicable to any physical instantiation of a qubit under investigation. For example, for polarization qubits, the $U_\pm$, $V_\pm$, and $W_\pm$ could be related to the horizontal/vertical, diagonal/anti-diagonal, and right/left circular polarization states, respectively.

The MUB defined above are over-complete, and a determination of detection probability in each of the six states is sufficient to reconstruct an unknown quantum state [9]. In principle, state reconstruction is

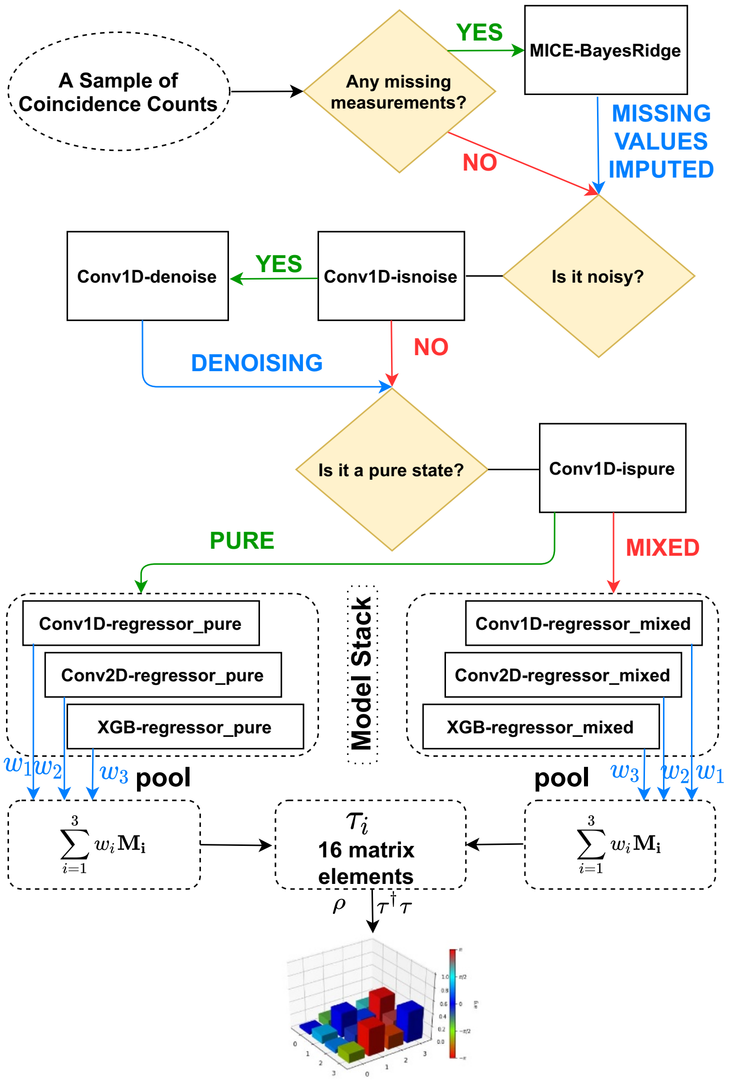


Figure 1. Machine learning pipeline for quantum state estimation.

possible using a more judiciously chosen set of measurement operators; for a single qubit, only four are required [23]. However, typical QST systems use the over-complete set defined by the MUB since it minimizes the statistical errors associated with estimating probability distributions from finite samples [23, 25]. Extensions of QST to n qubits can be made simply by determining the joint probabilities associated with all 6^n measurement combinations on all qubits [9].

We are now ready to summarize the process of QST on an n qubit system; to establish, through repeated measurement, the probabilities associated with all 6^n measurement combinations and then determine the quantum state most consistent with those results. The inversion of the measurement probabilities into a valid quantum state is the computationally expensive part of QST. Details about conventional methods for reconstructing the quantum state from measurement statistics using maximum likelihood estimation can be found in [9]. Alternatively, as in our previous work, machine learning systems can be trained on large amounts of simulated measurement results to perform the inversion with generally equivalent performance [6].

In standard quantum theory the measurement probability M associated with a given projector P is given by $M[P] = \mathrm{Tr}(\rho P)$. The aim of QST for n qubits, then, is to simultaneously and consistently invert a set of 6^n measurement results M to determine the state ρ. To reconstruct the two-qubit system we are interested in, we need to determine 36 joint probabilities. Each of these probabilities is associated with a projection operator of the following form

$$P = \begin{bmatrix} u_+ \otimes u_+ & u_+ \otimes u_- & u_- \otimes u_- & u_- \otimes u_+ & u_- \otimes w_+ & u_- \otimes w_- \\ u_+ \otimes w_- & u_+ \otimes w_+ & u_+ \otimes v_+ & u_+ \otimes v_- & u_- \otimes v_- & u_- \otimes v_+ \\ v_- \otimes v_+ & v_- \otimes v_- & v_+ \otimes v_- & v_+ \otimes v_+ & v_+ \otimes w_+ & v_+ \otimes w_- \\ v_- \otimes w_- & v_- \otimes w_+ & v_- \otimes u_+ & v_- \otimes u_- & v_+ \otimes u_- & v_+ \otimes u_+ \\ w_+ \otimes u_+ & w_+ \otimes u_- & w_- \otimes u_- & w_- \otimes u_+ & w_- \otimes w_+ & w_- \otimes w_- \\ w_+ \otimes w_- & w_+ \otimes w_+ & w_+ \otimes v_+ & w_+ \otimes v_- & w_- \otimes v_- & w_- \otimes v_+ \end{bmatrix}, \tag{2}$$

where $u_\pm = |U_\pm\rangle\langle U_\pm|$, $v_\pm = |V_\pm\rangle\langle V_\pm|$, and $w_\pm = |W_\pm\rangle\langle W_\pm|$. For this set of projective measurements the joint detection probability for a state ρ is given by

$$M[6i+j] = \mathrm{Tr}(\rho P[i,j]); \quad \text{for} \quad i,j = 0, 1, 2, 3, 4, 5. \tag{3}$$

Here the matrix has been unrolled into a 1×36 dimensional row vector and the sampling occurs by the row-major ordering of matrices.

Any density matrix ρ reconstructed from a set of measurement results M must be physically valid, that is, it must be a non-negative definite Hermitian matrix of trace one. To ensure this is the case, we do not predict a density matrix ρ directly from measurement results, and instead design our network so that the output (firing of 16 neurons) comprises the elements of the τ-matrix, which can be listed as $(\tau_0,\tau_1,\tau_2,\tau_3, \ldots, \tau_{15})$. These outputs are then rearranged to form a lower triangular matrix as given by

$$\tau = [\tau_0, \tau_1, \tau_2, \tau_3, \ldots, \tau_{15}] \rightarrow \begin{bmatrix} \tau_0 & 0 & 0 & 0 \\ \tau_4 + i\tau_5 & \tau_1 & 0 & 0 \\ \tau_{10} + i\tau_{11} & \tau_6 + i\tau_7 & \tau_2 & 0 \\ \tau_{14} + i\tau_{15} & \tau_{12} + i\tau_{13} & \tau_8 + i\tau_9 & \tau_3 \end{bmatrix}, \tag{4}$$

which always maps to a physically valid density matrix [33] by

$$\rho = \frac{\tau^\dagger \tau}{\mathrm{Tr}\left(\tau^\dagger \tau\right)}. \tag{5}$$

For training purposes it is also essential to be able to generate the τ matrix associated with a given density matrix ρ. This can be accomplished using the methods of [33] given by

$$\tau = \begin{bmatrix} \sqrt{\frac{Det(\rho)}{m_1^{00}}} & 0 & 0 & 0 \\ \frac{m_1^{01}}{\sqrt{m_1^{00} m_2^{00,11}}} & \sqrt{\frac{m_1^{00}}{m_2^{00,11}}} & 0 & 0 \\ \frac{m_2^{01,12}}{\sqrt{\rho^3 3}\sqrt{m_2^{00,11}}} & \frac{m_2^{00,12}}{\sqrt{\rho^{33}}\sqrt{m_2^{00,11}}} & \sqrt{\frac{m_2^{00,11}}{\rho^{33}}} & 0 \\ \frac{\rho^{30}}{\sqrt{\rho^{33}}} & \frac{\rho^{31}}{\sqrt{\rho^{33}}} & \frac{\rho^{32}}{\sqrt{\rho^{33}}} & \sqrt{\rho^{33}} \end{bmatrix}, \tag{6}$$

where m_1^{ij} for $i, j \in \{0, 1, 2, 3\}$, and $m_2^{pq,rs}$ ($p \neq r$ and $q \neq s$) for $p, q, r, s \in \{0, 1, 2, 3\}$ are the first and second minor of ρ, respectively.

Finally, we note that our metric for estimation accuracy between a generated matrix ρ and a target matrix ρ_0 is the fidelity, given by

$$F = \left| \mathrm{Tr} \sqrt{\sqrt{\rho_{pred}} \rho_{targ} \sqrt{\rho_{pred}}} \right|^2. \tag{7}$$

2.2. Simulation noisy measurement results

To simulate noisy measurement results, we introduce arbitrary rotations into the projective bases of P. This noise model is experimentally inspired and represents the difficulty of correctly determining and aligning measurement bases. Arbitrary rotations to the measurement bases are applied using the rotational operator $\mathcal{R}$ given by

$$\mathcal{R}(\vartheta, \varphi, \xi) = \begin{bmatrix} e^{i\varphi/2}\cos(\vartheta) & -ie^{i\xi}\sin(\vartheta) \\ -ie^{-i\xi}\sin(\vartheta) & e^{-i\varphi/2}\cos(\vartheta) \end{bmatrix}, \tag{8}$$

where I is the identity matrix. The variables ϑ, φ, ξ are sampled from the normal distribution with zero mean and σ^2 variance. Application of this noise model to M results in

$$M_{noise}[6i+j] = \mathrm{Tr}(\rho(I \otimes \mathcal{R})P[i,j](I \otimes \mathcal{R}^\dagger)). \tag{9}$$

Throughout this paper, the elements of $\mathcal{R}$ are treated as random variables and are sampled individually for each element i and j of M. Therefore, each of the 36 measurements in M has independent noise, meant to reflect how an experimentalist might individually align each bases measurement in M.

2.3. Generating random two-qubit states

So that our QST system applies to all possible input states, we train and test it against both pure and mixed states generated at random. Random pure states of two-qubits are created by generating Haar random 4×4 unitary matrices and taking the first column as the state's coefficients. Specifically, given a Haar random unitary $\mathcal{A}$ the accompanying pure state is

$$|\psi\rangle = \mathcal{A}_{00}|U_+U_+\rangle + \mathcal{A}_{10}|U_+U_-\rangle + \mathcal{A}_{20}|U_-U_+\rangle + \mathcal{A}_{30}|U_-U_-\rangle, \tag{10}$$

where $\mathcal{A}_{ij}$ represents the ith row and jth column of $\mathcal{A}$. Note that we add a tiny perturbation term ε (1×10^{-7}) to the simulated pure states as $\rho_{pure} = (1-\epsilon)|\psi\rangle\langle\psi| + \frac{\epsilon}{4}I$ to avoid the possible convergent issue under Cholesky decomposition [34].

Random mixed states are generated from the Ginibre ensemble [35] given by

$$G = N\big(0,1,[4,4]\big) + iN\big(0,1,[4,4]\big), \tag{11}$$

where $N\big(0,1,[4,4]\big)$ represents the random normal distribution of size of 4×4 with zero mean and unity variance. The random mixed state is extracted from this ensemble using

$$\rho_{\text{mix}} = \frac{GG^\dagger}{\text{Tr}(GG^\dagger)}, \tag{12}$$

where Tr represents the matrix trace.

2.4. Generating training sets

In order to train the models on our state estimation pipeline we start by randomly generating 1 million pure states, 1 million mixed states, and their corresponding tomography measurements. This yields design matrices, $X_{\text{noiseless}}$_pure, of size $1M \times 36$ for projective measurements from noiseless pure states, $X_{\text{noiseless}}$_mixed of size $1M \times 36$ for measurements from noiseless mixed states, and regression target matrices $Y_{\text{noiseless}}$_pure of size $1M \times 16$ of τ elements for pure states, and $Y_{\text{noiseless}}$_mixed of size $1M \times 16$ for mixed states. We split each of them into training, validation (hold-out), and test sets with ratios of 90% (900 K), 5% (50 K), and 5% (50 K), respectively.

Next, we generate noisy measurements. For that purpose, we generate 400 sets of noisy measurements per density matrix we created. The random rotation angles ϑ, φ, ξ for the unitary matrix given in equation (8) are randomly sampled to generate random matrices to create measurements using equation (9). For each of these sets of 400 measurements (400 random matrices) the angles ϑ, φ, ξ are sampled from the following distributions of mean zero, and standard deviation of σ; 100 of them from the normal distribution $\mathcal{N}(0,\sigma)$, 100 of them from the Laplace distribution $L(0,\sigma)$, 50 of them from brown noise $Br(0,\sigma)$, 50 of them from blue noise $Bl(0,\sigma)$, and 50 of them from pink noise $Pink(0,\sigma)$. Normal and Laplacian noise are sampled using *Numpy* [36], while colored noises are sampled by modifying the *colorednoise* package [37] following the recipe given in [38]. Next, we create another 9000 pure states and 9000 mixed states, separating each of them into three sets of 3000, then splitting these sets of 3000 s into separate training (90%, 2700), validation (5%, 150), and test sets (5%, 150). From each of these sets of 3000 states, we create 1.2M noisy measurements through 400 random unitary rotations such that we get 1080 K training, 60 K validation, and 60 K test measurements per set of 3000. We sample random angles with different standard deviations for each of these in 3 sets of 3000 by standard deviations of $\pi/24$, $\pi/12$, $\pi/6$ such that we generate 3.6 million noisy pure state measurements. We do the same to the three sets of 3000s for the mixed states to obtain 3.6 million noisy mixed state measurements. The respective training-validation-test sets of each get measurements from different quantum states to prevent over-fitting, e.g. the set of mixed states with noise $\pi/6$ has 1.2M measurements and is split into subsets of 1080 K on training, 60 K on validation, and 60 K on test sets, but each of these subsets contain the noisy measurements of different quantum states. These noisy measurement sets comprise two design matrices X_{allnoise} and X_{nonoise}. The former has 400 rows of noisy measurement samples of size 1×36 vertically stacked per density matrix, while the latter has 400 copies of the noiseless measurement stacked per density matrix such that their dimensions match. Also, for the τ matrix elements to be estimated, each set has the target variable matrix Y_{nonoise}. For regression, there are 400 copies of 1×16 τ samples stacked per density matrix. At the end we have generated 7.2M noisy measurements; 3.6M from pure states (1.2M from $\pi/24$, 1.2M from $\pi/12$, 1.2M from $\pi/6$), and 3.6 from mixed states (1.2M from $\pi/24$, 1.2M from $\pi/12$, 1.2M from $\pi/6$). These correspond to the following three matrices: X_{allnoise} of size 7.2$M \times 36$ with all original rows, X_{nonoise} of size 7.2$M \times 36$ with 18 K original rows (due to 9 K pure and 9 K mixed), and Y_{nonoise} of size 7.2$M \times 16$ with 18 K original rows.

Finally, we generate matrices for our classifier, which detects whether a state is pure is mixed. We vertically stack one million measurements from pure noiseless states, 3.6 million from pure noisy states, one million from noiseless mixed states, and 3.6 million from noisy mixed states to create the matrix X_{ispure}. The corresponding the target vector $\mathbf{t}_{\text{ispure}}$ has the first 4.6 million rows as zeros and the next 4.6 million rows as ones. Similarly, we vertically stack 1 million measurements from the pure noiseless stack with 1 million from the noiseless mixed states and stack them with 7.2 million from the noisy states to create the matrix X_{isnoise}. We generate a target label vector t_{isnoise} that has its first 2 million entries for noiseless states as zero and the next 7.2 million entries as ones.

2.5. Imputation with MICE-BR

Imputation is a statistical technique to infer missing values of input predictors from the rest of the data on hand. Single imputation using regression (Buck method) first fills the missing values with the given predictor's sample mean. It then fits a linear regression model for each predictor given predictors are highly correlated [39]. The overcomplete nature of state tomography with mutually unbiased bases, totaling 36 measurement probabilities for two-qubits, means the measurement results are strongly correlated for a wide array of input states. Therefore, linear regression models for each of 36 measurement probabilities using other values can be fitted to predict missing tomography measurements [40]. Despite the circular dependence, i.e. fitting a model to predict x_a using x_b, then fitting another for vice versa, this single imputation technique works remarkably well for such correlated variables [41]. In our case, it corresponds to fitting simple linear regression models (design matrix composed only of predictors as features) with the zeroth measurement, x_0, as the target variable while using the other 35 inputs as features, then fitting a model with x_1 as targets while using the rest of 35 measurements as features, with the pattern continuing. With fitted probabilistic linear models, stochastic regression imputation can sample imputation values from the imputation model's normal predictive distribution (e.g. a Gaussian on stochastic regression's predicted mean and variance) fit on features [41].

We employ a technique known as Multiple Imputation by Chained Equations (MICE), which runs multiple stochastic imputation models with different random seeds that can be used to sample and pool multiple values [42]. A single imputation algorithm *IterativeImputer* interatews over multiple imputations in a round-robin fashion [42] yet still converging to a single value, making it easily corrupted by noise [41]. By pooling values obtained from different random seeds, MICE is more robust to noise.

We fit four different Bayesian Ridge (BR) models to the concatenated training sets of noiseless pure (900 K$\times$ 36) and mixed (900 K$\times$ 36) state measurements (1.8 M) with the parameters $\alpha_1 = \alpha_2 = \lambda_1 = \lambda_2 = 10^{-6}$ under the *IterativeImputer* wrapper. The BR regressions we fit to predict measurements learn how to regularize the weight of other measurements from the data itself, preventing overfitting (e.g. if measurement 35 is not affecting measurement 0, it will have a smaller weight while regressing for 0). Then, during the *transform* stage, we use $t = 15$ iterations and four random seeds on the individual iterative imputations.

To assess training performances of algorithms that impute missing at random (MAR) [43] type data, Monte Carlo (MC) methods can be used to simulate the imposition of missing data into a complete set. As the imposition of MAR resembles a simple probit type of regression (i.e. logistic regression of whether the binary choice of a particular value is missing with a probability), missingness corresponds to random sampling from the data with respect to a probability density function; i.e. using random masks to turn off values [44]. The value to be estimated can be obtained through random masks shaped by low discrepancy sampling (quasi-random numbers) such as Halton sequences [45]. Our quasi-Monte Carlo simulations of the MICE performance involves generating masks that randomly turn off a given number of measurements using Halton sequences. For example, to simulate the case of a single measurement gone missing, we may randomly turn the 20th, 12th, 35th, 3rd (and so on) measurements for the first fourth tomography measurement samples (and so on) into *NaN* (not a number), then impute them using MICE. We take 10 K noiseless samples of measurements, create 50 masks of size 10 K$\times$ 36 to emulate one measurement missing, then 100 to emulate two, etc, increasing by 50 until reaching 500 masks (turning off ten measurements). Additionally, we use 500 masks to take MC estimates for any number of missing measurements higher than 10. We find that there is almost no noise up to 5 missing measurements. The MSE quadratically increases as demonstrated by the quadratic polynomial with coefficient $a = 1.9 \times 10^{-5}$ (without intercept or linear term) we fit in figure 2. However, as seen in figure 2, the more measurements are missing, the more statistical noise is introduced to the recovered measurements.

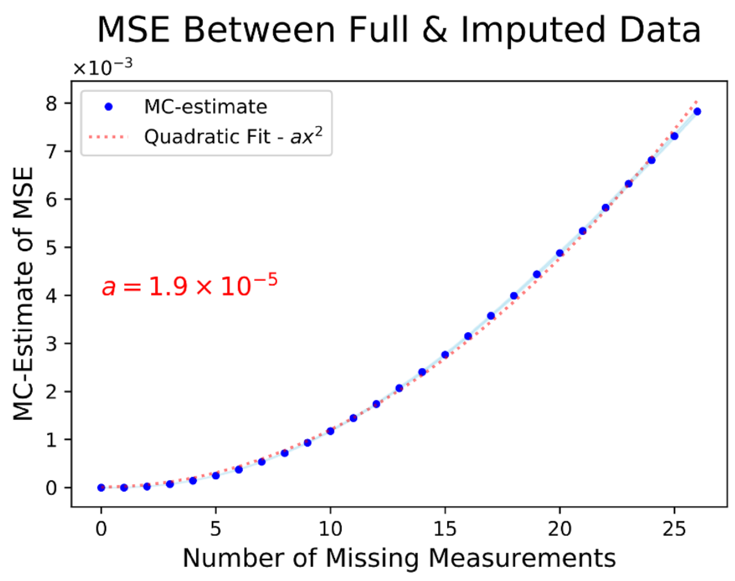


Figure 2. Noise induced by missing values imputation.

Having imputed missing measurements using low-level features of MICE-BR, though the contribution weight of each measurement on imputing another is learned from the data, we need more sophisticated algorithms for state estimation. Quantum state estimation, formulated as a regression problem, requires high-level features or high-level predictor segmentation. For that, we turn to the neural network type algorithms, then boosting trees.

2.6. Convolutional neural networks and auto-encoders

In comparison to the user-defined features (basis functions and kernels) of predictors supplied within the design matrix in the linear models, the main advantage of artificial neural networks (ANNs) is that they learn these features from the data by themselves [46]. The initial layer of ANNs take the bare input predictors, fit a multiple linear regression model plus a non-linear activation function (ReLu, tanh, etc) towards neurons of the next layer, and repeats this until reaching the output layer [46]. One specific type of neural network layer, a convolutional layer, enables learning and the application of complicated convolutional filters [47]. As the depth of a convolutional neural network (CNN) increases, so does the sophistication of the convolutional filters and features that can be used for challenging regression and classification tasks [48]. Auto-encoders (AE), a particular type of ANNs, has an equal amount of input predictor and output target dimensions [46]. The AE is composed of an encoder and a subsequent decoder portion with a signature bottleneck architecture, starting with neurons (feature weights) and feature-mappings (convolutional feature outputs) of higher dimensions, moving towards lower dimensions at the bottleneck, and finishing with higher dimensions again. This bottleneck forces the neural network to learn lower-dimensional, latent representations as the encoded features (encoder output) that can be reconstructed into higher-dimensional output targets by the decoder network [46]. As an instance of unsupervised learning, feeding a convolutional AE's input and output layers with input predictors (independent variables) enables the user to extract latent features later used in denoising, regression, and classification tasks from the data. Adding drop-out layers to these convolutional AEs, as an approximation to Bayesian NNs, further allows the user to weed out redundant features, which would lead to over-fitting, using the data alone [49]. For example, for the problem of fitting a model $h(t)$ into the data of a sophisticated, non-linear function $f(t)$, such as $y \sim f(t) = 2\sin(t)^3 - \log(t)^4$, supplying the predictors t to a convolutional AE enables the user to extract complicated features such as $\sin(t)$, $\log(t)$, $\exp(t)$, while the drop-out layer disables the redundant feature $\exp(t)$. When the user freezes the weights of the encoder portion that extracted features, and complement it with a simple regression ANN that takes the dependent, target variable y as the output, the added regressor portion only needs to learn the simple polynomial function of the previously extracted features. This avoids dealing with the short-comings of deep neural networks such as slow convergence and vanishing gradients [50, 51]. In the same spirit we use convolutional autoencoders to extract complicated features out of 36 projective measurements to later use them for the denoising, regression and classification tasks in our QST pipeline.

We train five different unsupervised auto-encoder models on Keras with Tensorflow backend [52]. We use a batch size of 256 and shuffle training data at each epoch for our models. First, by supplying a 900 K

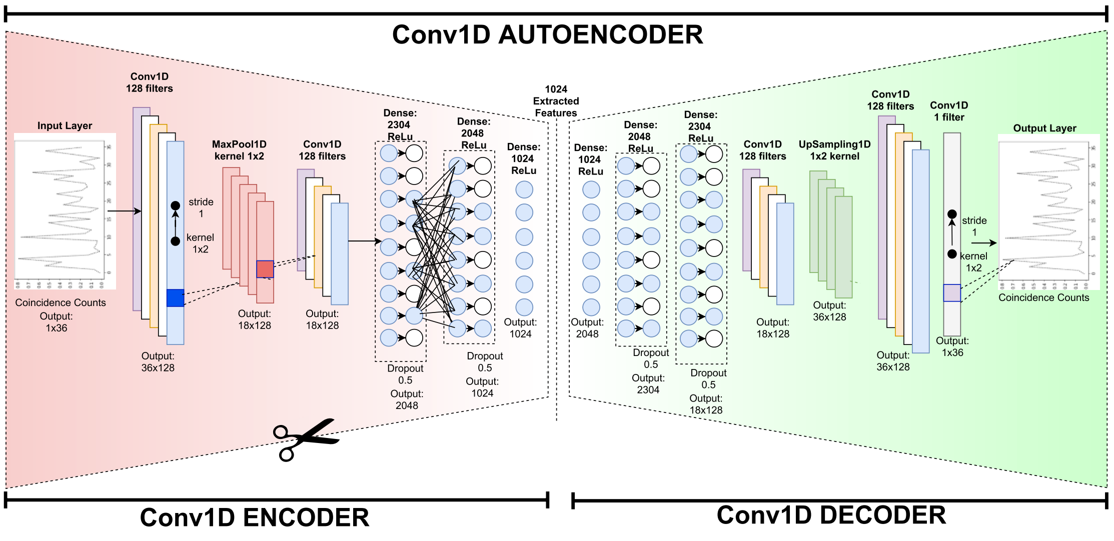


Figure 3. Our 1D convolutional auto-encoder design to be used for denoising and feature extraction for both regression and classification. The scissor icon is to denote the encoder portion is frozen, and cloned to be used in different neural networks. Our 2D convolutional auto-encoder works in an analogous fashion.

noiseless pure state measurement training set as both the input and the output, while validating on 50 K (validation set of noiseless pure states) again as both input and output, we train two auto-encoders that use Conv1D-autoencoder and Conv2D-autoencoder architecture using the MSE loss function. For clarity we include a visualization of the Conv1D-autoencoder in figure 3. The Conv2d-autoencoder is not pictured but functions analogously. We train both of these AEs using one epoch of Adam optimizer with learning rate $\eta = 0.003$, two epochs of Adam optimizer with learning rate $\eta = 0.001$, one epoch of standard gradient descent (SGD) with learning rate $\eta = 0.001$, batch size of 256, and executing early stopping when the validation set MSE reaches 4×10^{-4}. As a result we obtain the intermediate models *Conv1D-AE-noiseless_pure* and *Conv2D-AE-noiseless_pure*. Then, we take two clones of *Conv1D-AE-noiseless_pure* and one clone of *Conv2D-AE-noiseless_pure*. The clone of *Conv2D-AE-noiseless_pure*, and the first clone of *Conv1D-AE-noiseless_pure* are re-trained (on top of their previous training as weight initialization) using a 900 K noiseless mixed state measurement training set as both the input and the output, while validating on their 50 K validation set as the input and output, as the intermediate models *Conv2D-AE-noiseless_mixed* and *Conv1D-AE-noiseless_mixed*. These two models are both trained using one epoch of Adam optimizer with learning rate $\eta = 0.001$, one epoch of SGD with learning rate $\eta = 0.001$, and early stopping set for a validation MSE of 4×10^{-4}. The second clone of *Conv1D-AE-noiseless_pure* is also trained further by adding noiseless mixed state data on top of the previous data as an intermediate model *Conv1D-AE-noiseless_both*. Concatenating 900 K from training sets and 50 K from validation sets of the noiseless pure and mixed state measurements *Conv1D-AE-noiseless_both* model takes these $1.8M$ measurements as the training data (both as input and output) while validating on the 100 K. *Conv1D-AE-noiseless_both* is then trained with the exact same epochs and learning rates as the *Conv2D-AE-noiseless_mixed*.

Second, we transfer the encoder portion of the *Conv2D-AE-noiseless_both*, use its extracted latent features and freeze its weights to a NN that uses the Conv1D-Classifier architecture given in figure 4 to a model we named *Conv1D-isnoise* in the flow chart given in figure 1. The classifier portion to be trained is initialized randomly, while the initial encoder layers transferred are not trained. We also clone this model into another classifier named *Conv1D-ispure*. *Conv1D-isnoise* is trained using the training set of the matrix X_{isnoise} given the above as the input, and the training set of the vector labeled t_{isnoise} as the output features while validating on their respective validation sets. Similarly, *Conv1D-ispure* is trained using the training set of the matrix X_{ispure} and the training set of the vector labeled t_{ispure} as the output features while validating on their respective validation set. We use *binary cross-entropy* as the loss function. Due to unsupervised pre-training using auto-encoders, both *Conv1D-isnoise* and *Conv1D-ispure* classifiers obtains a binary prediction performance of 0.99999 F-score [53] and AUC [54] for the training, validation, and test sets in just one epoch of training with Adam optimizer and learning rate $\eta = 0.001$. The training performance details for these classification models are given in the table.

Third, we take another clone of pre-trained *Conv1D-AE-noiseless_both* to train a supervised denoising auto-encoder model given in figure 1 as *Conv1D-denoise*. Both in the training and validation stages, *Conv1D-denoise* takes 400 noisy measurements per density matrix as the input and tries to map it to the 400

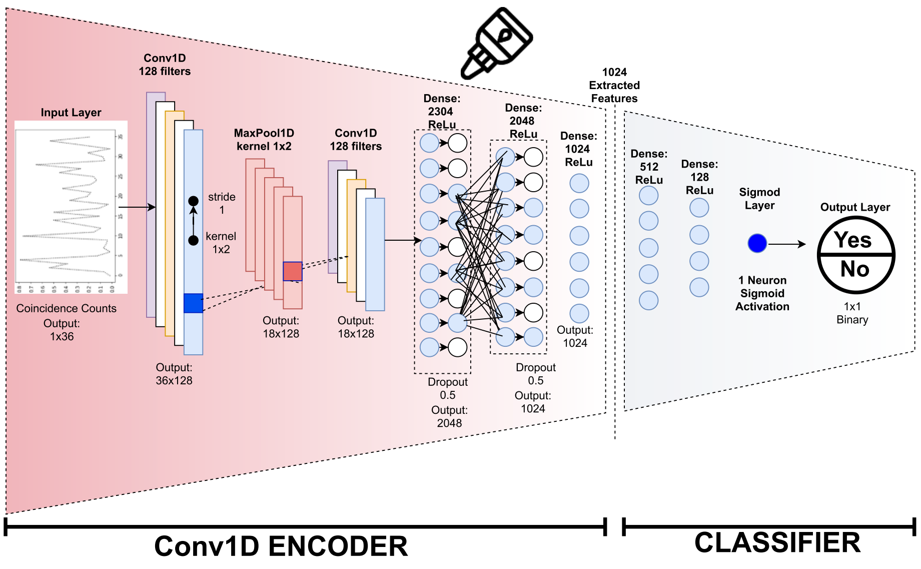


Figure 4. Our NN design for classification.

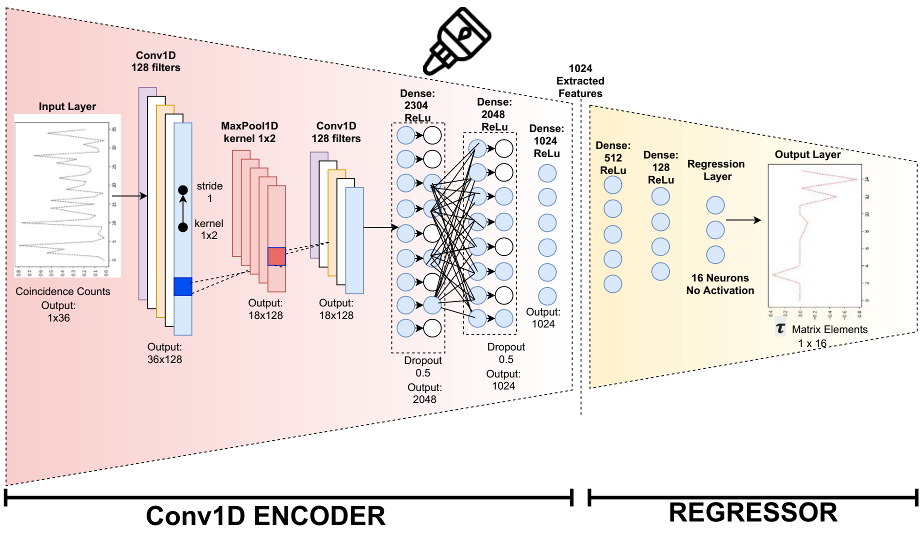


Figure 5. Conv1D-regressor design.

Table 1. 1D *ConvNet classifier* performances on train-validation-test sets of generated data.

	Train *F*-score	Validation *F*-score	Test AUC
Conv1D-isnoise	0.9999	0.9999	0.9999
Conv1D-ispure	0.9999	0.9999	0.9999

copies of the original, noiseless measurements from the said density matrix. We use the training set of X_{allnoise} given above as the input and X_{nonoise} as the output features while validating on their respective validation sets. After training for five epochs using Adam optimizer with learning rate $\eta = 0.001$, and two epochs with SGD with learning rate $\eta = 0.001$, we halt the training when the early stopping criteria of training and validation set MSE of 10^{-3} is reached. Bench-marking on the test set, we also observe a testing set denoising performance of 10^{-3} MSE.

Fourth, the encoder portions of *Conv1D-AE-noiseless_pure* and *Conv2D-AE-noiseless_pure*, and *Conv1D-AE-noiseless_mixed* and *Conv2D-AE-noiseless_mixed* are transferred to the *Conv1D-regressor_pure*, *Conv2D-regressor_pure*, *Conv1D-regressor_mixed*, *Conv2D-regressor_mixed* models, with the one dimensional designs shown in figure 5, with analogous designs for the two-dimensional case. Again only the 1024 latent features extracted by the pre-trained AEs encoder portion are used, without further training the

Table 2. 1D and 2D *ConvNet regressor* performances on train-validation-test sets of generated data.

	Train MSE	Validation MSE	Validation fidelity	Test fidelity
Conv1D-regressor_pure	5×10^{-4}	4×10^{-4}	1.0	1.0
Conv2D-regressor_pure	7×10^{-4}	6×10^{-4}	0.995	0.995
Conv1D-regressor_mixed	5×10^{-4}	4×10^{-4}	1.0	1.0
Conv2D-regressor_mixed	8×10^{-4}	7×10^{-4}	0.997	0.997

transferred encoder portion for all the four models. These encoder portions are horizontally stacked with the regressor NN portions to be trained. *Conv1D-regressor_pure* and *Conv2D-AE-noiseless_pure* are trained using the training set of $X_{\text{noiseless}}$_pure as the input predictors, and $Y_{\text{noiseless}}$_pure as the output regression targets, while validating on their respective validation sets. Similarly we train *Conv1D-regressor_mixed* and *Conv2D-AE-noiseless_mixed* using the training set of $X_{\text{noiseless}}$_mixed as the input predictors, and $Y_{\text{noiseless}}$_mixed as the output regression targets, while validating on their respective validation sets. We train these four models using three epochs of Adam optimizer with learning rate $\eta = 0.001$, and two epochs of SGD with learning rate $\eta = 0.001$. Training of those that meet the early stopping criterion of validation set with MSE 4×10^{-4} are halted before the end of epochs. By computing their quantum fidelities, we found that Conv1D based models have an average fidelity of 1.0 and a standard deviation of 0.0 in their respective test sets, and Conv2D ones are also nearly unity. The training performance details for these regression models are given in the table below.

2.7. XGB-regression

In contrast to externally imposing features of predictors (as basis functions in design matrix) to data in linear models, or learning these advanced features from data as in neural networks, tree based algorithms work with a completely different formalism of stratifying predictor space into simple regions [55]. The classifying and regression trees (CART) algorithm fragments the predictor space by splitting nodes of decision trees with respect to a gain metric [56]. It does so by using the recursive binary splitting algorithm to grow a tree that stops to yield a split when information gain is maximized for classification, and residue sum of squares (MSE) is minimized for regression problems by the said split [55]. Despite CART's ability to agnostically handle non-linear data, it possesses low predictive power. The contemporary algorithms compensate for this by using *boosting*. Instead of hard fitting one complicated CART, boosting sequentially grows simple trees to the errors of the former such that it learns slowly from the data [57]. In the regression context, this means fitting a tree model on training data, then fitting another one on the residual of the former, and so on. However, when these new models are fit into the residuals, residuals of residuals, etc, they are not hard to fit to get new residuals. Instead, they are scaled by a shrinkage factor to force the algorithm to learn many different models slowly. The state of the art for tree-based algorithms, Extreme Gradient Boosting Algorithm (XGB), unlike regular Gradient Boosting Algorithms that use CART, uses its unique tree structure [58]. XGB first fits a leaf and computes a residual. Then fits an XGB tree onto that residual but scales it with learning rate η to compute residual of a residual, and so on. For example, for the previously given case $y \sim f(t) = 2\sin(t)^3 - \log(t)^4$, XGB first fits a leaf using a base score (default is 0.5), meaning it maps all the independent, input variables to the base score while predicting the dependent variable y. Then calculates the residuals between the predicted value (base score) and the actual y values to fit a unique XGB tree onto them. This tree is utilized to make predictions, and these predictions are scaled by η to compute residuals of residuals, and so on.

We use two models, one for noiseless pure states *XGB-regressor_pure*, and one for noiseless mixed states *XGB-regressor_mixed*, that use the multiple target XGB regression for the state estimation architecture given in figure 6. The former uses the training and validation sets of $X_{\text{noiseless}}$_pure as the input, and $Y_{\text{noiseless}}$_pure as the output. The latter uses those of $X_{\text{noiseless}}$_mixed and $Y_{\text{noiseless}}$_mixed. The native Python API of XGB does not support multiple target regression. Still, it has features like *DMatrix* data format for parallelization and computational speed up, automatic early stopping, and XGB's inherent cross-validation toolset. Therefore, instead of using XGB's Scikit-learn API and Scikit-learn's wrapper for multiple target regression, *MultipleOutputRegression*, to combine single XGB regression models from the start at the training and model-selection (i.e. hyper-parameter tuning) stages, we choose to train and tune 16 different models in XGB's native Python API, then combine them with *MultipleOutputRegression* at the prediction (testing) stage.

We start the hyper-parameter tuning by a grid search of parameters using 5-fold cross-validation (CV) on the training set [55]. We split the training set into five subsets, and for five turns, we holdout one of these subsets for evaluation while training on rest and take the mean of these subset evaluations at the end. We, in

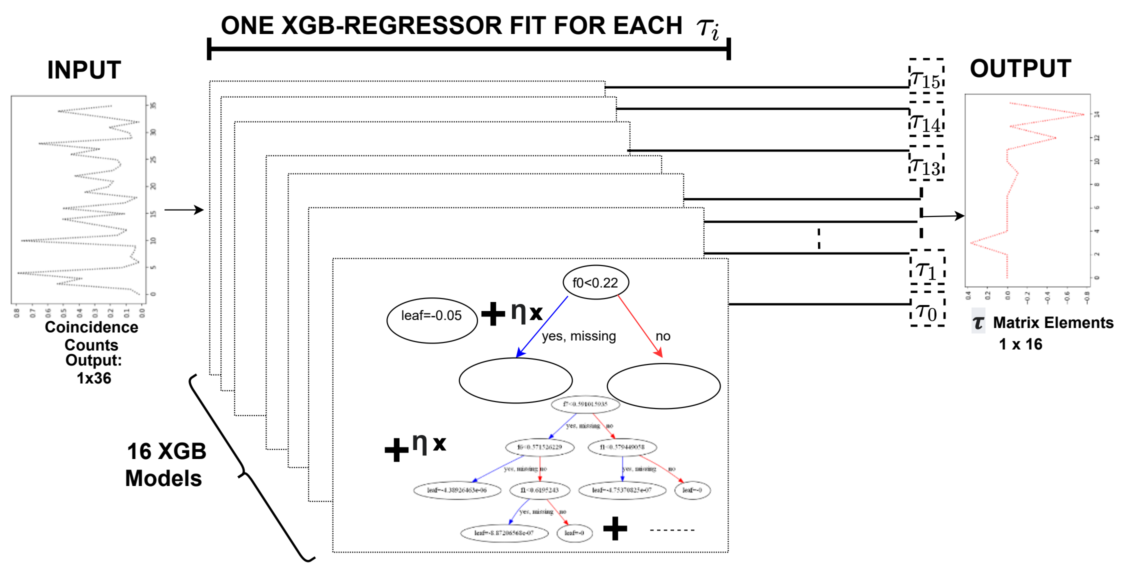


Figure 6. Our multiple target XGB regression design uses 16 separate XGB regression models fit to each of 16 τ matrix elements, using 36 tomography measurements.

the beginning, cross-validate while fitting a model only on the τ_0 using 36 tomography measurements by choosing the *objective* parameter in native Python API *reg:linear* for regression. We first create a 2D grid of parameters *max_depth* and *min_child _weight* ranging from 9 to 12 and 5 to 8. Scanning through this grid while keeping learning rate $\eta = 0.3$, early stopping rounds of 10, and everything else at default. We obtain the least root-mean-squared-error (RMSE, $\sqrt{\text{MSE}}$) scores as the evaluation metric. We find the best improvement in CV scores for a *max_depth* of 9 and *min_child _weight* of 6 with a RMSE of 0.099 (9.8×10^{-3} MSE). The former parameter determines the maximum depth of XGB regression trees that are allowed to fit, and the latter determines the minimum sum of weights (Hessians) needed per 'child' (when we do a split). Instead of optimizing all the hyper-parameters together, we save these results from the first 2D grid search and use them in our upcoming grid search.

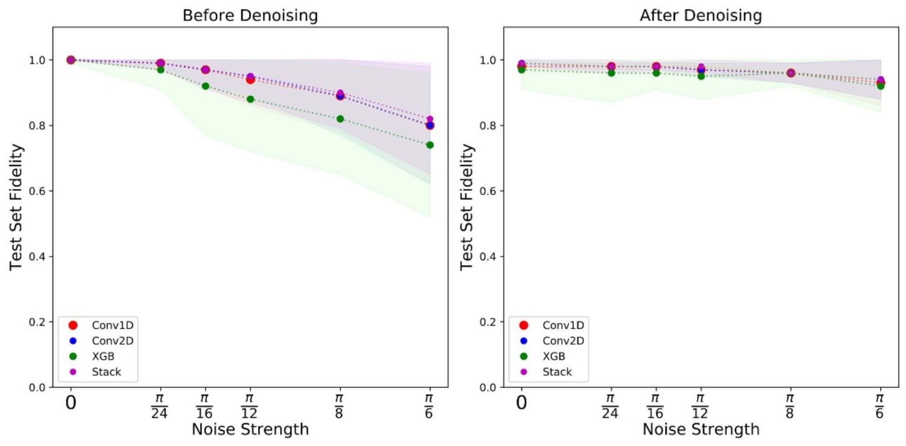
This time we create another 2D grid ranging from 0.7 to 1 for the both dimensions to scan the *subsample* and *colsample_bytree* parameters. The first parameter is to control what ratio of training instances (rows) are randomly sampled to grow trees (e.g. 0.5 uses half) to control over-fitting, and the latter does the same thing, but for columns. We found the ideal value for both of them to be 1.0, leading to a CV score of 3×10^{-2} RMSE (9×10^{-4} MSE). Last, we search for optimal Ridge (l_2) and Lasso (l_1) penalty strengths, λ and α, by scanning through a square grid with both sides ranging from 0 to 1. The former regularization has the effect of shrinking the values of predictors (features) to a small but nonzero value. Simultaneously, the latter leads to parsimony (sparsity) due to completely turning weights for some features. We found that $\lambda = 0$ and $\alpha = 0.4$ leads to a CV RMSE of 9×10^{-3} (8.1×10^{-5} MSE). Finally, using the previously optimized hyper-parameters, by executing a 1D grid search for learning rate η ranging from 0.01 to 0.3, we found 0.1 to be optimal with CV RMSE of 7×10^{-3} (4.9×10^{-5} MSE). We found no improvement by running the same model selection process through a 5-fold CV for the other 15 τ elements.

Using the optimal parameters found above, we fit 32 XGB regression models, 16 for noiseless pure states, and 16 noiseless mixed states. We set the number of rounds (a.k.a., number of boosts, the equivalent of epochs for XGB) to 2000. However, due to early stopping, when validation set RMSE does not improve for ten rounds, training halts. We found most of the 32 models achieved very high accuracy (validation set MSE of the order $\sim 10^{-6}$), even more significant than neural nets, in less than 50 rounds (a couple of seconds in a laptop with a GPU). Some τ elements were harder to fit, and it took them around 200 rounds to fit with a validation set MSE of the order $\sim 10^{-3}$. Having trained 16 models for pure and 16 for mixed states, we combine their 'boosters' (XGB models) within the Sci-kit learn *MultipleOutputRegression* wrapper. Both the model for noiseless pure states and the mixed states achieve training, validation, and test set MSEs on the order $\sim 10^{-4}$, and an average fidelity 1.0 with standard deviation 0.0.

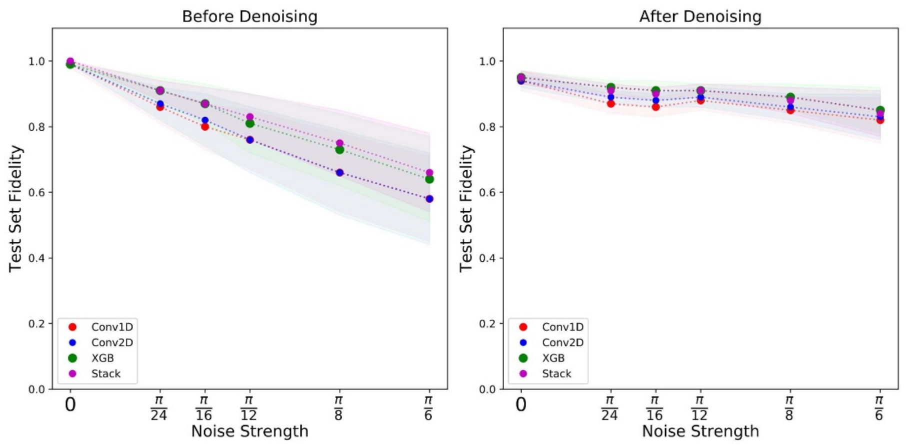
The computational speed needed to fine-tune the model for extreme accuracy via reducing feedback time and the ability to work with missing values and noise, makes XGB a perfect candidate to work alongside and fix the mistakes of these other algorithms. In the QST context, there is an incentive to use a meta-model of different types of models, each with their advantage on interpreting tomography measurements, to tackle

Table 3. *XGB regressor* performances on train-validation-test sets of generated data.

	Train MSE	Validation MSE	Validation fidelity	Test fidelity
XGB-regressor_pure	9×10^{-5}	8×10^{-5}	1.0	1.0
XGB-regressor_mixed	6×10^{-5}	5×10^{-5}	1.0	1.0



(a) Pure states



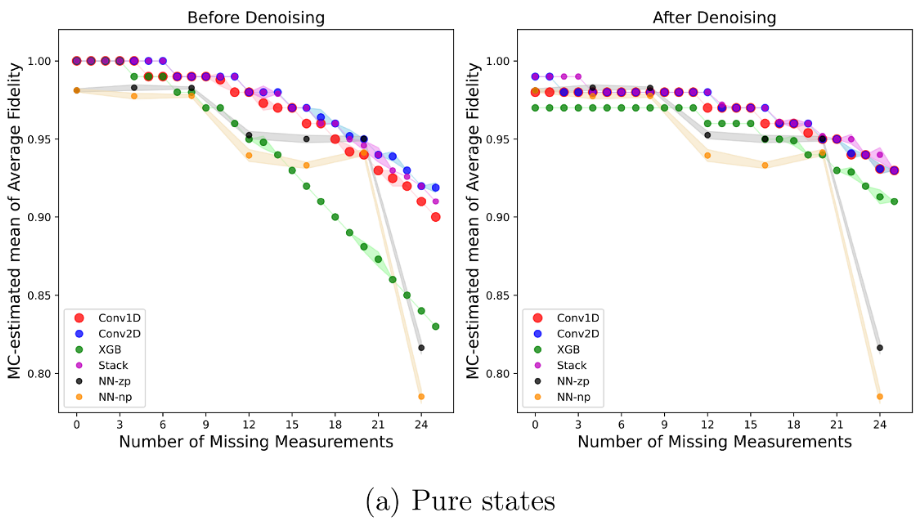
(b) Mixed states

Figure 7. Test set fidelity vs. noise strength for (a) pure and (b) mixed states before and after denoising. Solid dots denote the average fidelity of the test set, while the colored hues denote the variance (standard deviation) of the fidelity. CNNs perform better for pure states, while XGB performs better for mixed states. Stacking and denoising increases the performance in all cases. Denoising when no noise is present decreases prediction accuracy.

both the physical noise related to the measurement basis alignment and the statistical noise induced by the imputation procedure.

2.8. Meta-model

Ensemble learning is the method of using an ensemble of machine learning algorithms as a committee to make predictions [59]. The boosting method we utilized is an ensemble learning method using subsequent models, where the latter models are trained on the former's output. However, ensemble methods are not restricted to such dependent models. Stacked generalization is an ensemble learning approach that uses machine learning models that are trained using a training set as *level-0* generalizers, and learn a *level-1* meta-model on a hold-out (validation) data set to combine the prediction of the low-level models for predictive performance improvements that originators of the algorithm deemed 'black art' [60]. Low-level model predictions could be averaged or pooled as a weighted sum by giving models with higher accuracy higher weights, or their combinations for pooling can be learned by the meta-learner altogether [61]. Stacked generalization could use the same or different models as the *level-0* base-learners, including neural networks, to boost the performance [62]. Due to the resulting performance increase, researchers using stacking



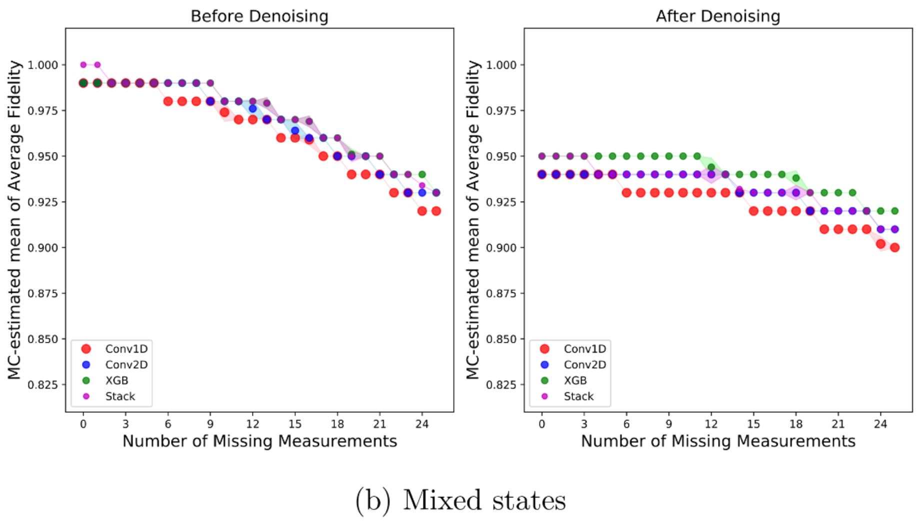
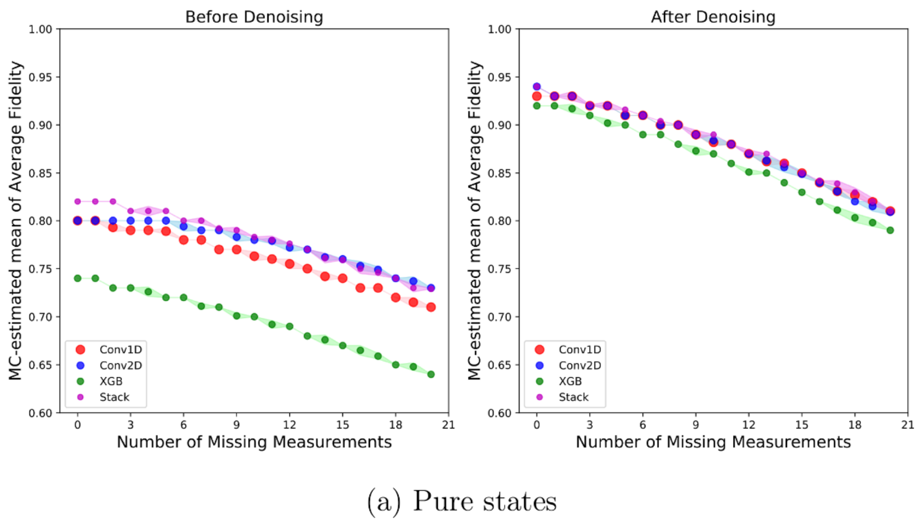


Figure 8. MC estimation for the estimation of noiseless: (a) pure and (b) mixed states after k measurements went missing on test set. Solid dots denote MC estimate mean for the average fidelity, while the hue denotes the variation of this average fidelity for different masks. We found stacking of models helps for both pure and mixed states. Passing a sample with imputed missing measurements through a denoising AE, which is originally trained using simulated experimental noise from basis rotations not the statistical imputation noise, before QST models in a pipeline improves reconstruction performance. Here 'NN-zp' and 'NN-np' denote the neural networks with zero-padding and no-padding results on pure states from our previous work for comparison [6].

techniques have won awards in machine learning competitions such as The Netflix Prize [63]. In our case, we found that pure and mixed states can not be regressed together for state estimation, but they instead need their own set of machine learning models. We discovered that Conv1D, Conv2D, and XGB models have their own strengths and weaknesses while processing noisy and incomplete data due to how they interact with features. Thus, we need one meta-learner of stacked base models for both pure and mixed states. This way, these stacked *level-0* state estimation models cover each others' mistakes when the *level-1* meta-model pools them.

We train two linear regression models to find the pooling weights for pure and mixed state models. In order not to over-fit, stacked generalization training is done using the validation set predictions [60]. Concatenating a 50 K validation set of noiseless pure states with 180 K noisy states to get 230 K measurements, resulting in 230 K τ predictions from *Conv1D-regressor_pure*, *Conv2d-regressor_pure* and *XGB-regressor_pure* models and concatenating them, we obtain a 230 K $\times$ 48 input matrix. We use this input matrix as the design matrix for the linear model. As the dependent target variables, we use the ground-truth 230 K$\times$ 16 τ matrix elements. Fitting them for pure states, we obtain meta-model pooling weights $w_1 = w_2 = 0.4$ and $w_3 = 0.2$. This means the *Conv1D* and *Conv2D* models generalize better for the noisy pure states than *XGB*, yet all these models still cover each others' faulty predictions for better generalization performance. On the other hand, when we fit a linear meta-model for the mixed states, we found the pooling weights to be $w_1 = w_2 = 0.16$ while $w_3 = 0.67$, meaning the *XGB* model generalizes better for the mixed states.



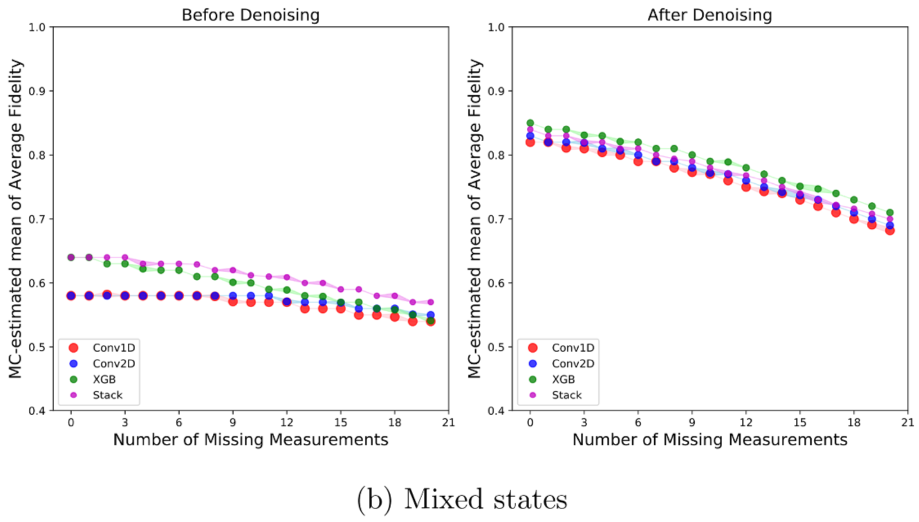


Figure 9. MC estimation for the estimation: (a) pure and (b) mixed states after imputation using noisy ($\pi/6$) measurements. Solid dots denote MC estimate mean for the average fidelity, while the hue denotes the variation of this average fidelity for different masks. We found stacking of models helps for both pure and mixed states. Passing a sample that contains both simulated experimental noise from basis rotations and imputed missing measurements through a denoising AE, which is originally trained using fiber noise not the statistical imputation noise, before QST models in a pipeline improves reconstruction performance.

3. Results

We start by assessing the performance of state estimation models with respect to noise with or without denoising. We generate one set of 10 K noiseless measurements and five sets of 10 K noisy measurements of noise strengths $\pi/24$, $\pi/16$, $\pi/12$, $\pi/8$, $\pi/6$ for pure states, and the same for mixed states. Plugging them into our QST models for pure and mixed states and their respective stacks, we, as expected, observed a decrease in predictive performance with the increasing noise strength, as shown in figure 7. We found that CNN based models are more resilient to noise for pure states, while the XGB based model has higher predictive power for mixed states, and the stacking of respective pure and mixed state models helps both. We observed in figure 7 that with the increase of noise, not only does the average fidelity decrease (solid dots), but also the variance on fidelity (colored hues) increases, meaning noise on some states are tolerated even less. However, turning on the denoiser has a stabilizer effect. It increases the average fidelities that were reduced with increasing noise and shrank the test set fidelity variances (hues in the figure). We also observe that denoising when no noise is present decreases performance; hence it underlines the importance of detecting which measurement samples are noisy (*Conv1D-isnoise*).

Next, we use Monte Carlo simulations via Halton sequences, explained in previous sections, to estimate the QST model performances when k measurements are missing and recovered using *MICE-BR*. We again create random masks to emulate missing measurements (50 masks for a single measurement missing, 100 for double that, etc), calculate average test set fidelity for the measurements recovered by *MICE-BR* after they were turned off for each mask, and compute the Monte Carlo mean of the average fidelity, and the MC standard deviation of the average fidelity.

First, we use the MC average fidelity estimation scheme for the generated test sets of noiseless pure and mixed state measurements, see figure 8. We found that the denoiser, which was trained using data coming from physical noise due to measurement basis misalignment during detection, not the MICE statistical noise, does not seem to improve imputation noise in terms of MSE between imputed and the original data (thus not included in that figure). Despite that, we found *Conv1D-denoise* actually improves prediction accuracy without improving MSE between X_{original} and $X_{\text{recovered}}$. Even when $k = 26$ measurements are missing (using only ten measurements), we still recover average fidelity values above 90%. We found that our current approach that combines imputation, stacking and denoising performs better than our previous approach that combines zero-padding or no-padding with 2D convolutional neural networks (figure 8) [6]. In our previous work, we had two neural networks per k missing measurements (e.g. for $k = 1$ we replaced 36th measurement with zero for zero-padding and used 35-dimensional input vectors for no-padding), our current approach not only has the advantage of compensating for $\binom{36}{k}$ combination of those missing measurements, but it also outperforms the previous models in terms of fidelity, especially when more than 20 measurements are missing. When we turn on the physical noise for these imputed measurements, the effects of stacking and the denoising AE are more pronounced for both the pure and the mixed states, as seen in figure 9.

4. Discussion

In conclusion, we built a pipeline of machine learning models for quantum state estimation using projective measurements. Projective measurements coming to the pipeline have their missing values imputed, directed to denoising if they are detected to be noisy, and their quantum states estimated using different models depending on whether they are deemed to be pure or mixed states.

Unlike our previous work, we decoupled the handling of noise and the treatment of missing measurements from the models' training. This approach allows us to generalize our previous results to the case of any number and combinations of missing, noisy measurements, rather than the limited special cases.

Data availability statement

The data that support the findings of this study are available from the corresponding authors on reasonable request.

Acknowledgments

This material is based upon work supported by, or in part by, the Army Research Laboratory and the Army Research Office under Contract/Grant Numbers W911NF-19-2-0087 and W911NF-20-2-0168. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Author contributions statement

O D developed the neural networks and ran all simulations. R T G and B T K conceived of and led the project. O D, R T G, S L and B T K wrote the manuscript. All authors contributed to the discussions and interpretations of the results.

Conflict of interests

The authors declare no competing interests.

ORCID iDs

Onur Danaci https://orcid.org/0000-0002-7320-8297
Sanjaya Lohani https://orcid.org/0000-0003-0699-0669

References

[1] Carleo G, Cirac I, Cranmer K, Daudet L, Schuld M, Tishby N, Vogt-Maranto L and Zdeborová L 2019 Machine learning and the physical sciences *Rev. Mod. Phys.* **91** 045002

[2] Carleo G, Nomura Y and Imada M 2018 Constructing exact representations of quantum many-body systems with deep neural networks *Nat. Commun.* **9** 1–11

[3] Lennon D T, Moon H, Camenzind L C, Liuqi Y, Zumbühl D M, Briggs G A D, Osborne M A, Laird E A and Ares N 2019 Efficiently measuring a quantum device using machine learning *npj Quantum Inf.* **5** 79

[4] Nautrup H P, Delfosse N, Dunjko V, Briegel H J and Friis N 2019 Optimizing quantum error correction codes with reinforcement learning *Quantum* **3** 215

[5] Kalantre S S, Zwolak J P, Ragole S, Wu X, Zimmerman N M, Stewart M D and Taylor J M 2019 Machine learning techniques for state recognition and auto-tuning in quantum dots *npj Quantum Inf.* **5** 1–10

[6] Lohani S, Kirby B, Brodsky M, Danaci O and Glasser R T 2020 Machine learning assisted quantum state estimation *Mach. Learn.: Sci. Technol.* **1** 035007

[7] Zimmermann B, Mueller T, Meineke J, Esslinger T and Moritz H 2011 High-resolution imaging of ultracold fermions in microscopically tailored optical potentials *New J. Phys.* **13** 043007

[8] Qian X and Shuqi X 2018 Neural network state estimation for full quantum state tomography (arXiv: 1811.06654 [quant-ph])

[9] Altepeter J B, Jeffrey E R and Kwiat P G 2005 Photonic state tomography *Adv. At. Mol. Opt. Phys.* **52** 105–59

[10] Smolin J A, Gambetta J M and Smith G 2012 Efficient method for computing the maximum-likelihood quantum state from measurements with additive Gaussian noise *Phys. Rev. Lett.* **108** 070502

[11] Bo Q, Hou Z, Li Li, Dong D, Xiang G and Guo G 2013 Quantum state tomography via linear regression estimation *Sci. Rep.* **3** 3496

[12] Hou Z *et al* 2016 Full reconstruction of a 14-qubit state within four hours *New J. Phys.* **18** 083036

[13] Torlai G, Mazzola G, Carrasquilla J, Troyer M, Melko R and Carleo G 2018 neural-network quantum state tomography *Nat. Phys.* **14** 447

[14] Carrasquilla J, Torlai G, Melko R G and Aolita L 2019 Reconstructing quantum states with generative models *Nat. Mach. Intell.* **1** 155–61

[15] Torlai G and Melko R G 2018 Latent space purification via neural density operators *Phys. Rev. Lett.* **120** 240503

[16] Xin T, Sirui L, Cao N, Anikeeva G, Dawei L, Jun Li, Long G, and Zeng B2018 Local-measurement-based quantum state tomography via neural networks (arXiv: 1807.07445 [quant-ph])

[17] Palmieri A M, Kovlakov E, Bianchi F, Yudin D, Straupe S, Biamonte J and Kulik S 2019 Experimental neural network enhanced quantum tomography (arXiv: 1904.05902 [cond-mat, physics: quant-ph])

[18] Lohani S, Knutson E M, Zhang W and Glasser R T 2019 Dispersion characterization and pulse prediction with machine learning *OSA Contin.* **2** 3438–45

[19] Czerwinski A 2020 Quantum state tomography of four-level systems with noisy measurements (arXiv: 2101.00016)

[20] Lukens J M, Law K J H and Bennink R S 2020 A Bayesian analysis of classical shadows (arXiv:2012.08997)

[21] Ahmed S, Muñoz C S, Nori F and Kockum A F2020 Quantum state tomography with conditional generative adversarial networks (arXiv: 2008.03240)

[22] Ahmed S, Muñoz C S, Nori F and Kockum A F 2020 Classification and reconstruction of optical quantum states with deep neural networks (arXiv: 2012.02185)

[23] Řeháček J, Englert B-G and Kaszlikowski D 2004 Minimal qubit tomography *Phys. Rev.* A **70** 052321

[24] Zhu H 2014 Quantum state estimation with informationally overcomplete measurements *Phys. Rev.* A **90** 012115

[25] Wootters W K and Fields B D 1989 Optimal state-determination by mutually unbiased measurements *Ann. Phys.* **191** 363–81

[26] Buzek Vır, Drobny G, Derka R, Adam G and Wiedemann H1998 Quantum state reconstruction from incomplete data (quant-ph/9805020)

[27] Thiago O, Cesário A T and Vianna R O2011 Variational quantum tomography with incomplete information by means of semidefinite programs *Int. J. Mod. Phys.* C **22** 1361–72

[28] Gonçalves D S, Lavor C, Gomes-Ruggiero M A, Cesário A T, Vianna R O and Maciel T O 2013 Quantum state tomography with incomplete data: maximum entropy and variational quantum tomography *Phys. Rev.* A **87** 052140

[29] Teo Y S, Stoklasa B, Englert B-G, Řeháček J and Hradil Zěk 2012 Incomplete quantum state estimation: a comprehensive study *Phys. Rev.* A **85** 042317

[30] Teo Y S, Zhu H, Englert B-G, Řeháček J and Hradil Zěk 2011 Quantum-state reconstruction by maximizing likelihood and entropy *Phys. Rev. Lett.* **107** 020404

[31] Gross D, Liu Y-K, Flammia S T, Becker S and Eisert J 2010 Quantum state tomography via compressed sensing *Phys. Rev. Lett.* **105** 150401

[32] Flammia S T, Gross D, Liu Y-K and Eisert J 2012 Quantum tomography via compressed sensing: error bounds, sample complexity and efficient estimators *New J. Phys.* **14** 095022

[33] Daniel F V James P G Kwiat W J and White A G 2001 Measurement of qubits *Phys. Rev.* A **64** 052312

[34] Higham N J 1990 Analysis of the Cholesky decomposition of a semi-definite matrix *Reliable Numerical Computation* (Oxford: Oxford University Press) pp 161–85

[35] Forrester P J and Nagao T 2007 Eigenvalue Statistics of the real Ginibre ensemble *Phys. Rev. Lett.* **99** 050603

[36] Harris C R *et al* 2020 Array programming with Numpy *Nature* **585** 357–62

[37] Patzelt F 2019 *colorednoise.py—A Python Package to Generate Colored Noise* Python Package

[38] Timmer J and Koenig M 1995 On generating power law noise *Astron. Astrophys.* **300** 707–10

[39] Buck S F 1960 A method of estimation of missing values in multivariate data suitable for use with an electronic computer *J. R. Stat. Soc.* B **22** 302–6

[40] Wang S X and Kanter G S 2009 Robust multiwavelength all-fiber source of polarization-entangled photons with built-in analyzer alignment signal *IEEE J. Sel. Top. Quantum Electron.* **15** 1733–40

[41] Little R J A and Rubin D B 2019 *Statistical Analysis with Missing Data* vol 793 (New York: Wiley)

[42] van Buuren S and Groothuis-Oudshoorn K 2011 MICE: multivariate imputation by chained equations in R *J. Stat. Softw.* **45** 1–68

[43] Rubin D B 1976 Inference and missing data *Biometrika* **63** 581–92

[44] Lang K M and Little T D 2014 The supermatrix technique: a simple framework for hypothesis testing with missing data *Int. J. Behav. Dev.* **38** 461–70

[45] Halton J H 1960 On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals *Numer. Math.* **2** 84–90
[46] Goodfellow I, Bengio Y and Courville A 2016 *Deep Learning Book* vol 521 (Cambridge, MA: MIT Press) p 800
[47] Simard P Y, Steinkraus D and Platt J C *et al* 2003 Best practices for convolutional neural networks applied to visual document analysis *ICDAR* vol 3
[48] Krizhevsky A, Sutskever I and Hinton G E 2012 Imagenet classification with deep convolutional neural networks *Advances in Neural Information Processing Systems* pp 1097–105
[49] Gal Y and Ghahramani Z 2016 Dropout as a Bayesian approximation: representing model uncertainty in deep learning *Int. Conf. on Machine Learning* pp 1050–9
[50] Tang C and Monteleoni C 2018 Demystifying overcomplete nonlinear auto-encoders: fast SGD convergence towards sparse representation from random initialization
[51] Sagheer A and Kotb M 2019 Unsupervised pre-training of a deep LSTM-based stacked autoencoder for multivariate time series forecasting problems *Sci. Rep.* **9** 1–16
[52] Chollet F *et al* 2015 Keras a deep learning API written in Python (https://keras.io)
[53] Sasaki Y *et al* 2007 The truth of the F-measure (www.cs.odu.edu/~mukka/cs795sum09dm/Lecturenotes/Day3/F-measure-YS-26Oct07.pdf)
[54] Fawcett T 2006 An introduction to ROC analysis *Pattern Recognit. Lett.* **27** 861–74
[55] James G, Witten D, Hastie T and Tibshirani R 2013 *An Introduction to Statistical Learning* vol 112 (Berlin: Springer)
[56] Breiman L, Friedman J, Stone C J and Olshen R A 1984 *Classification and Regression Trees* (Boca Raton, FL: CRC Press)
[57] Freund Y, Schapire R and Abe N 1999 A short introduction to boosting *J. Japan. Soc. Artif. Intell.* **14** 1612
[58] Chen T, Tong H, Benesty M, Khotilovich V and Tang Y 2015 XGBoost: extreme gradient boosting *R package version 0.4–2* pp 1–4
[59] Brown G 2010 *Ensemble Learning* (Boston, MA: Springer US) pp 312–20
[60] Wolpert D H 1992 Stacked generalization *Neural Netw.* **5** 241–59
[61] Ting K M and Witten I H 1999 Issues in stacked generalization *J. Artif. Intell. Res.* **10** 271–89
[62] Zhou Z-H, Jianxin W and Tang W 2002 Ensembling neural networks: many could be better than all *Artif. Intell.* **137** 239–63
[63] Bell R M, Koren Y and Volinsky C 2007 The Bellkor solution to the Netflix prize *KorBell Team's Report to Netflix*