Machine learning the spectral function of a hole in a quantum antiferromagnet

Jackson Lee, Matthew R. Carbone, and Weiguo Yin

Phys. Rev. B **107**, 205132 — Published 16 May 2023

DOI: 10.1103/PhysRevB.107.205132

Machine-learning the spectral function of a hole in a quantum antiferromagnet

Jackson Lee,[1,2] Matthew R. Carbone,[3] and Weiguo Yin[2,*]

[1]*Physics and Computer Science Department, Rutgers University, New Brunswick, New Jersey 08854, USA*
[2]*Condensed Matter Physics and Materials Science Division, Brookhaven National Laboratory, Upton, New York 11973, USA*
[3]*Computational Science Initiative, Brookhaven National Laboratory, Upton, New York 11973, USA*

(Dated: May 3, 2023)

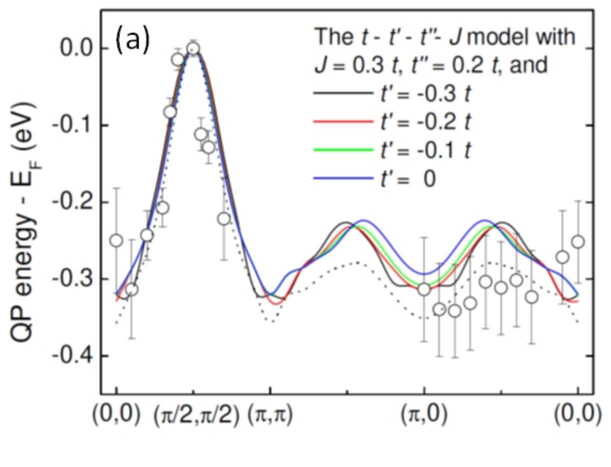
Understanding charge motion in a background of interacting quantum spins is a fundamental problem in quantum many-body physics. The most extensively studied model for this problem is the so-called t-t'-t''-J model, where the determination of the parameter t' in the context of cuprate superconductors is challenging. Here we present a theoretical study of the spectral functions of a mobile hole in the t-t'-t''-J model using two machine learning techniques: K-nearest Neighbors regression (KNN) and a feed-forward neural network (FFNN). We employ the self-consistent Born approximation to generate a dataset of about 1.3×10^5 spectral functions. We show that for the forward problem, both methods allow for the accurate and efficient prediction of spectral functions, allowing for e.g. rapid searches through parameter space. Furthermore, we find that for the inverse problem (inferring Hamiltonian parameters from spectra), the FFNN can, but the KNN cannot, accurately predict the model parameters using merely the density-of-state. Our results suggest that it may be possible to use deep learning methods to predict materials parameters from experimentally measured spectral functions.

I. INTRODUCTION

Understanding charge motion in a background of interacting quantum spins has been considered an essential first step in search for the mechanisms of superconductivity in many unconventional materials ranging from cuprates[1–3] to iron-based superconductors[4,5] and to twisted bilayer graphene.[6] This topic has recently received renewed interest thanks to recent novel experiments using ultracold atoms in optical lattices, as they provide an essentially perfect realization of the Fermi-Hubbard model, with site-resolved imaging ability.[7–15] The most extensively studied model for this problem is the so-called t-J-type model.[16] Its applicability to cuprates was established by comparing the model study results with various experiments, most notably angle-resolved photoemission spectroscopy (ARPES), which can specifically yield the spectral function of holes introduced by photoemission of electrons.[17–21] The single-hole problem corresponds to photoemission from an undoped Mott insulator, such as $\mathrm{Sr_2CuO_2Cl_2}$ (a parent compound of cuprates), where besides the nearest-neighbor hole hopping parameter (t), the second and third nearest-neighbor hopping parameters (t' and t'') are found to be necessary to reproduce the correct quasiparticle dispersion relation $E(\mathbf{k})$.[22–30] While the need for longer hopping parameters is justified by first-principles analysis of the in-crystal overlapping of electronic wave functions,[27,31] it was uncovered[27,28] that the determination of t' via fitting $E(\mathbf{k})$ is inconclusive because $E(\mathbf{k})$ could be insensitive to t' varying from 0 to $-0.3t$ [see Fig. 1(a)]. This is a relevant problem since the value of t' was shown to correlate with the superconducting transition temperature at optimal doping[31] and affect phase competition[32] in the cuprates.

Another drawback of this traditional approach to predicting model parameters is that $E(\mathbf{k})$ is generally derived from low-energy spectral peaks. However, it is difficult to resolve $E(\mathbf{k})$ when the quasiparticle spectral weight is small,[33,34] which is a common phenomenon in systems close to a non-Fermi liquid state, specifically near the high-energy edge of the quasiparticle band in cuprates, resulting in large error bars [see Fig. 1(a)]. It is thus highly desirable if the model parameters can be predicted by studying the full energy range of the spectral functions directly [see Fig. 1(b)]. Extension from fitting $E(\mathbf{k})$ to treating the whole spectral function $A(\mathbf{k}, \omega)$ means a dramatic increase in the total amount of data that needs to be processed. Here we use machine learning (ML) methods to address this outstanding problem in the field of strongly correlated electron systems and high-temperature superconductivity.

Machine learning plays a huge role when cataloguing or processing large amounts of data in general.[35] It is uniquely able to identify important patterns and correlations that might otherwise be missed, especially in large datasets. Recently, it has emerged as an important computational tool across disciplines in the physical sciences.[36] For example, in particle physics, ML played an instrumental role in the discovery of the Higgs boson.[37] In astrophysics, ML techniques have been used to study photometric redshifts,[38] cluster membership of galaxies,[39] and exoplanet transit detection.[40] In materials and molecular science, ML is heralding in a "second computational revolution",[41] helping predict crystal structures,[42] calculate material properties,[43–45] and accelerate first-principles calculations.[46–50] In condensed matter physics, ML was used to find phase transition temperatures,[51] catalogue snapshots of strongly correlated electronic states,[10] infer fundamental physical information from model systems,[52] efficiently sample configurations in many-body systems,[53] predict phases of



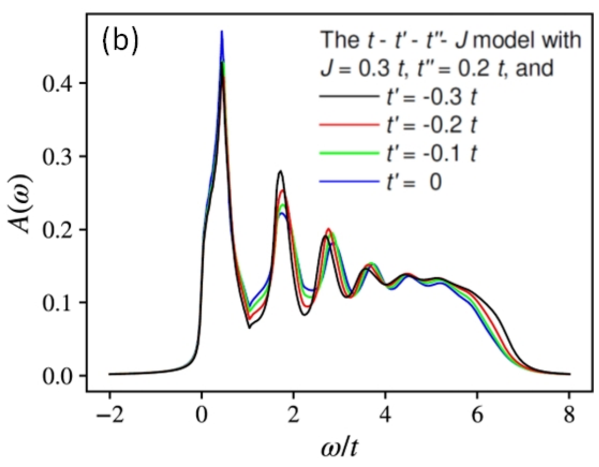


FIG. 1. The t' dependence of (a) the quasiparticle band dispersion $E(\mathbf{k})$ compared with experimental data (open circles), reproducing Fig. 4 in Ref. 27 with permission, and (b) the density of states $A(\omega) \propto \sum_{\mathbf{k}} A(\mathbf{k},\omega)$, where the first broad peaks around the Fermi level (zero energy) are almost identical for a wide range of t' but the other peaks could be used to resolve t'.

a non-exactly solvable model when trained on data of a solvable model,[54] and predict impurity spectral functions.[55]

In this paper, we show how ML can be used to predict and understand spectral functions in the t-t'-t''-J model by studying both the *forward* problem of predicting spectral functions from a given set of model parameters[55–57] $\{t,\ t',\ t'',\ J\}$ and the *inverse* problem of predicting the model parameters from spectral functions. We point out that the forward problem is in principle a matter of low-dimensional data interpolation, because an arbitrary (e.g. homogeneous dense) grid of values of the independent variables can be made in principle for the forward problem. On the other hand, the inverse problem of machine learning spectral functions involves significantly more generalization for two reasons: The first one is physically fundamental, namely spectral functions must satisfy certain sum rules. This means that in general, no single spectral function will fall fully inside the envelope created by another two. It is thus highly likely that parts of a spectral function stay outside all the spectral functions in the training dataset [cf. Fig. 1(b)]. In addition, an arbitrary (e.g. homogeneous dense) grid of values of the independent variables cannot be made for the inverse problem, because an arbitrarily constructed spectral function that does not satisfy the required sum rules has no physical meaning and no corresponding model parameters. The second reason is that of dimensionality, as the inverse problem is a matter of *high*-dimensional data generalization on the order of hundreds of inputs, as opposed to three for the forward problem on a desirable grid. We will show that these intrinsic differences between the forward and inverse problems make a classical ML algorithm and a deep neural network ML method both reasonable choices for the forward problem, but that their performances are quite different for the inverse problem. Our results showcase that deep neural networks could accurately handle such inverse problems.

The rest of this paper is organized as follows: Section II describes the t-t'-t''-J model, the used ML methods, and how we obtain the needed dataset for training, validation, and testing. Section III A presents a preliminary examination of the data using Principal Component Analysis (PCA), primarily to help determine what linear correlations may be present in the data. Section III B addresses the *forward* problem. Section III C addresses the *inverse* problem of predicting the model parameters t'/t, t''/t, and J/t from spectral functions. Section III D introduces an algorithm to find the value of t. Finally, the results and discussions are summarized in Section IV.

II. METHODS

A. Hamiltonian and spectral functions

The t-t'-t''-J model is described by the following Hamiltonian:[21]

$$H = -\left(t \sum_{\langle i,j\rangle_1,\sigma} + t' \sum_{\langle i,j\rangle_2,\sigma} + t'' \sum_{\langle i,j\rangle_3,\sigma}\right)\left(\tilde{c}^{\dagger}_{i\sigma}\tilde{c}_{j,\sigma} + h.c.\right) + J \sum_{\langle i,j\rangle_1} \mathbf{S}_i \cdot \mathbf{S}_j \qquad (1)$$

in the standard notation of the constrained fermionic operators: $\tilde{c}^{\dagger}_{i\sigma}$ creates an electron with the spin index σ (either $\uparrow$ or $\downarrow$) at site i—with the constraint of no double occupancy at any site—and $\tilde{c}_{i,\sigma}$ annihilates it. The spin operators $\mathbf{S}_i$ expressed in the matrix form are given by $(\mathbf{S}_i)_{\sigma\sigma'} = \frac{1}{2}\sum_{\sigma\sigma'} \tilde{c}^{\dagger}_{i\sigma}\hat{\tau}_{\sigma\sigma'}\tilde{c}_{i\sigma'}$ where $\hat{\tau} = (\hat{\tau}^x, \hat{\tau}^y, \hat{\tau}^z)$ are the 2×2 Pauli matrices. The angle brackets denote the first ($\langle i,j\rangle_1$), second ($\langle i,j\rangle_2$), and third ($\langle i,j\rangle_3$)

neighbor sites, respectively. Thus, the J term describes the Heisenberg interaction between nearest-neighboring quantum spins; the t, t', and t'' terms describe the electron hopping to nearest, second nearest, and third nearest sites, respectively.

The angle-resolved spectral function of a doped hole with momentum $\mathbf{k}$ and energy ω is given by

$$A(\mathbf{k},\omega) = -\frac{1}{\pi}\,\mathrm{Im}\,G(\mathbf{k},\omega), \tag{2}$$

with the retarded Green's function of the single hole being

$$G(\mathbf{k},\omega) = \lim_{\eta\to 0^+}\langle\Psi_0|\tilde{c}^\dagger_{\mathbf{k}\sigma}\frac{1}{\omega+i\eta-H+E_0}\tilde{c}_{\mathbf{k}\sigma}|\Psi_0\rangle, \tag{3}$$

where E_0 and $|\Psi_0\rangle$ are the ground-state energy and wave function of the undoped system, respectively, thus $H|\Psi_0\rangle = E_0|\Psi_0\rangle$. Or equivalently,

$$A(\mathbf{k},\omega) = \sum_\nu |\langle\nu|\tilde{c}_{\mathbf{k}\sigma}|\Psi_0\rangle|^2\delta(\omega - E_\nu + E_0), \tag{4}$$

where $|\nu\rangle$ is an eigenstate of H with one less electron and E_ν is the corresponding eigen-energy satisfying $H|\nu\rangle = E_\nu|\nu\rangle$, as the Dirac delta function $\delta(\omega)$ is related to a Lorentzian by $\delta(\omega) = \lim_{\eta\to 0^+}\frac{1}{\pi}\frac{\eta}{\omega^2+\eta^2} = \lim_{\eta\to 0^+} -\frac{1}{\pi}\mathrm{Im}\,\frac{1}{\omega+i\eta}$.

The angle-integrated spectral function is given by

$$A(\omega) = \frac{1}{N}\sum_{\mathbf{k}} A(\mathbf{k},\omega), \tag{5}$$

where $N = \sum_{\mathbf{k}} 1$ is the number of lattice sites. $A(\omega)$ is also called the density of states (DOS). To obtain the DOS in the normal procedure of theoretical calculations, one needs to have first calculated out $A(\mathbf{k},\omega)$ using Eq. (4) for a dense mesh of $\mathbf{k}$ points, and then sum the results over $\mathbf{k}$ using Eq. (5). This implies that if DOS can be accurately predicted from known DOS data, a significant speedup in evaluating DOS can be achieved, e.g., a four-orders-of-magnitude speedup compared with the normal procedure using a 100×100 $\mathbf{k}$-mesh. More interestingly, we will explore whether the model Hamiltonian parameters can be accurately predicted by machine learning the DOS $A(\omega)$, which is relevant to x-ray photoemission (XPS), or $A(\mathbf{k},\omega)$ with a fixed $\mathbf{k}$, which is relevant to laser-based ARPES where the $\mathbf{k}$ points are most accessible near the zone center $\mathbf{k} = 0$. The sum rules governing the spectral functions are $\int_{-\infty}^{+\infty} A(\mathbf{k},\omega)d\omega = \int_{-\infty}^{+\infty} A(\omega)d\omega = 1$.

B. Dataset generation

To obtain the dataset for use in our ML approach, we use the self-consistent born approximation (SCBA) to calculate Green's function of a hole in the t-t'-t''-J model[3,58–66] (see Appendix A for details). This approximation produces quantitatively accurate results for the hole Green's function compared with exact diagonalization on small systems[67] and Monte Carlo simulations.[68]

We note that although the Hamiltonian has four parameters (t, t', t'', J), all the data can be scaled with respect to t, e.g.

$$A(\mathbf{k},\omega) \to A(\mathbf{k}, a\,\omega)/a \quad \text{for} \quad t \to a\,t. \tag{6}$$

where a is an arbitrary positive real number. Setting t as the energy unit ($t = 1$) reduces the ML complexity by one dimension, which is of significant advantage in high-throughput computation and big data management. Thus, the Green's functions are generated in a grid of $t' \in [-0.5, 0.5]$, $t'' \in [-0.5, 0.5]$ and $J \in [0.2, 1.0]$, with each parameter sampled on a 51-point uniform grid.

For each combination of t', t'', and J, the calculation of the Green's function $G(\mathbf{k},\omega)$ is performed by using a 128×128 mesh for the $\mathbf{k}$ points, $\omega \in [-6, 6]$ with the step (i.e., energy resolution) being 0.01, and $\eta = 0.01$. Then, $\eta = 0.1$ is used to broaden the resulting spiky DOS and a uniform grid of 301 ω points is used to sample the DOS. Therefore, our dataset for the DOS consists of $51^3 = 132,651$ pairs $\left(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\right)$, with $\mathbf{x}^{(i)} = (t', t'', J)^{(i)}$ being the 3-dimensional vector representation of the ith model parameter set and $\mathbf{y}^{(i)} = (A(\omega_1), A(\omega_2), \ldots, A(\omega_{301}))^{(i)}$ the corresponding 301-dimensional vector representation of the DOS. For the forward problem, $\mathbf{x}^{(i)}$ are the input and $\mathbf{y}^{(i)}$ are the output. For the inverse problem, the definitions of $\mathbf{x}^{(i)}$ and $\mathbf{y}^{(i)}$ are switched, i.e., $\mathbf{x}^{(i)} = (A(\omega_1), A(\omega_2), \ldots, A(\omega_{301}))^{(i)}$ and $\mathbf{y}^{(i)} = (t', t'', J)^{(i)}$. We then randomly partitioned the dataset into an 80/10/10 training ($\mathbb{T}$), validation ($\mathbb{V}$), and testing $\mathcal{T}$ split. Here we use the computationally generated testing sets to demonstrate without any ambiguity that the ML methods work well for the present baseline problems.

C. Machine Learning Methods

Training a ML model consists of an optimization procedure in which a loss function encoding the difference between predicted and ground-truth outputs is minimized on a training set. In addition, a set of hyperparameters of the ML model is tuned during cross-validation to achieve high accuracy on the validation set. Hyperparameters are untrained parameters that include, but are not limited to, training time, network architecture and activation functions. Ultimately, final results are presented on the testing set in order to provide an unbiased estimate of model performance. Here we use the total mean squared error (MSE) as the loss function. Given the training set $\mathbb{T} = \left\{\left(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\right)\right\}$ of size $|\mathbb{T}|$, i.e., $i = 1, 2, 3, ..., |\mathbb{T}|$, for an n-dimensional input vector $\mathbf{x}^{(i)}$, the corresponding ground-truth output is a m-dimensional vector $\mathbf{y}^{(i)}$; if the ML model predicts $\hat{\mathbf{y}}^{(i)}$, then the individual MSE for that training example is given by

$$L^{(i)} = \frac{1}{m}\sum_{j=1}^{m}\left|\hat{y}_j^{(i)} - y_j^{(i)}\right|^2, \tag{7}$$

and the total MSE score of the ML method is given by

$$L = \frac{1}{|\mathbb{T}|}\sum_{i=1}^{|\mathbb{T}|} L^{(i)}. \tag{8}$$

We now introduce the two ML methods used in this work: K-nearest neighbors (KNN) and feed-forward neural network (FFNN).

1. *K-nearest neighbors*

The KNN algorithm predicts an output via a nearest neighbor search.[69] With a training set $\mathbb{T} = \left\{\left(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\right)\right\}$, the KNN algorithm finds the closest k points in the input parameter space. Then, a weighted average is taken over the outputs of the k neighbors to predict the output $\hat{\mathbf{y}}$ of a new input vector $\mathbf{x}$ by

$$\hat{\mathbf{y}} = \sum_{i\in \mathrm{NN}(\mathbf{x})} w^{(i)}(\mathbf{x})\mathbf{y}^{(i)}, \tag{9}$$

where $\mathrm{NN}(\mathbf{x})$ indicates the k nearest neighbors to $\mathbf{x}$. Here the weights $w^{(i)}$ is given by the inverse Euclidean distance[70]

$$w^{(i)}(\mathbf{x}) = \frac{\left|\mathbf{x}-\mathbf{x}^{(i)}\right|^{-\alpha}}{\sum_{j\in\mathrm{NN}(\mathbf{x})}\left|\mathbf{x}-\mathbf{x}^{(j)}\right|^{-\alpha}}, \quad i \in \mathrm{NN}(\mathbf{x}). \tag{10}$$

The optimized hyperparameters learned from training are $k = 9, \alpha = 5$ for the forward problem and $k = 9$, $\alpha = 3$ for the inverse problem.

2. *Feed-Forward Neural Network*

A neural network is a ML algorithm that implements multiple repeated blocks of linear predictions followed by the application of non-linear functions. In this work, we use feed-forward neural networks (FFNN), which consist of several layers of artificial neurons and is defined by how the layers are implemented and connected. Here we use a fully-connected FFNN in which neurons between adjacent layers are fully connected.[71] For example, a 3-layer FFNN is illustrated in Fig. 2. The architecture of a fully connected FFNN is primarily defined by the size of each layer, and the layer-by-layer one-way *activation* is given by $\mathbf{a}_l = f_l(W_l\mathbf{a}_{l-1} + \mathbf{b}_l)$ where $\mathbf{a}_l$ is the n_l-dimensional vector output of the lth layer, f_l is the activation function, W_l is an $n_l \times n_{l-1}$ matrix of weights (n_l is the number of neurons in layer l), and $\mathbf{b}_l$ is a vector of biases. Among them, W_l and $\mathbf{b}_l$ are learned during

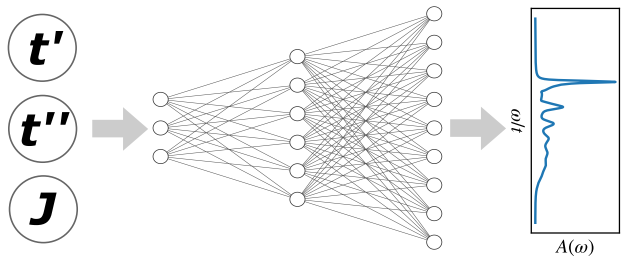


FIG. 2. A fully connected neural network, applied to the forward problem of predicting a DOS given t', t'', J. The neural network takes in a 3 dimensional vector representing t', t'', J, and outputs a 301 dimensional vector representing the predicted DOS.

training. For both the forward and inverse problems, we train the neural networks for a fixed length of time after which the validation loss changes little over time for various different FFNN architectures (30 minutes here,[72] cf. Fig. 9), using the rectified linear unit (ReLU) activation function $f_l(x) = \max(0, x)$ and Adam optimizer.[73]

III. RESULTS AND DISCUSSION

A. Principal component analysis

To analyze the quality of our dataset and visualize the potential of applying ML algorithms to the data, we performed the following principal component analysis (PCA)[74] (see Appendix B for details).

1. *Full DOS data*

We proceed with PCA of the dataset in which $\mathbf{y}^{(i)} = (A(\omega_1), A(\omega_2), \ldots, A(\omega_{301}))^{(i)}$ and $\mathbf{x}^{(i)} = (t', t'', J)^{(i)}$, where $i = 1, 2, 3, \ldots, 51^3$. Following Eq. (B2), we show the projected (reduced-dimensional) data vector in Fig. 3, and color each point $(z_1, z_2)^{(i)}$ by the value of $t'^{(i)}$, $t''^{(i)}$, and $J^{(i)}$, respectively, producing three subplots. All results look quite structured (ear like) and the color gradients are smooth. This suggests that the input parameters can be continuously mapped to spectral functions, making ML algorithms well suited for the forward problem. Furthermore, this suggests that the inverse problem of mapping spectral functions to input parameters is feasible.

2. *Using the first peak of DOS*

In comparison, the traditional method for predicting the Hamiltonian parameters is to fit the quasiparticle band $E(\mathbf{k})$ derived from the low-energy peak of the spectral function $A(\mathbf{k}, \omega)$, resulting a difficulty determining

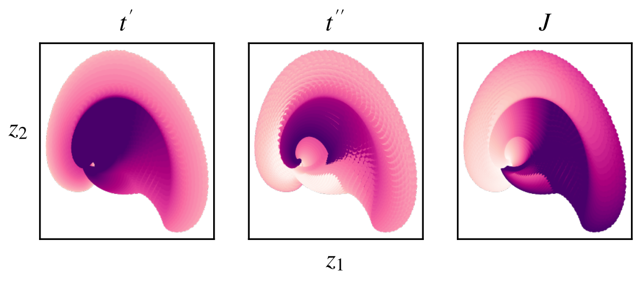


FIG. 3. 2D visualization of the DOS spectra projected into the first two principal components $(z_1, z_2)^{(i)}$. The color maps of the three subplots are determined by the values of t', t'', and J, respectively. The horizontal and vertical axes represent the first and second principal components, respectively.

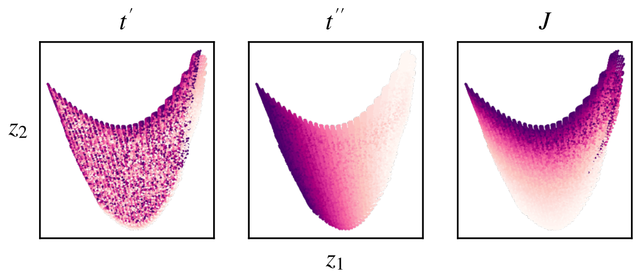


FIG. 4. 2D visualization of the lorentzian parameters (obtained from fitting the first peaks of the DOS spectra) projected into the first two principal components $(z_1, z_2)^{(i)}$. The color maps of the three subplots are determined by the values of t', t'', and J, respectively.

t'.[27,28] To visualize this problem with PCA, we use the Lorentzian

$$f(x) = \frac{A\gamma^2}{(x - x_0)^2 + \gamma^2} \tag{11}$$

to fit the first peak of every DOS spectrum considered in the inverse problem, resulting in a feature dataset in which now $\mathbf{y}^{(i)} = (A, x_0, \gamma)^{(i)}$. Then, we redo PCA and show the color maps of the first two principal components in Fig. 4. We see that while the t'' and J plots have smooth gradients, the t' plot contains much more scattering data, demonstrating the difficulty in resolving t' by using only the first-peak information.

B. The forward problem

KNN.— We first trained a KNN for the forward problem (see Appendix C 1 for details) and found that with the optimized hyperparameters $k = 9$ and $\alpha = 5$, the KNN was able to produce DOS that are almost visually identical to the SCBA results. The worst percentiles of prediction for the testing set, in terms of mean squared error score, are shown in Fig. 5. We note that even for

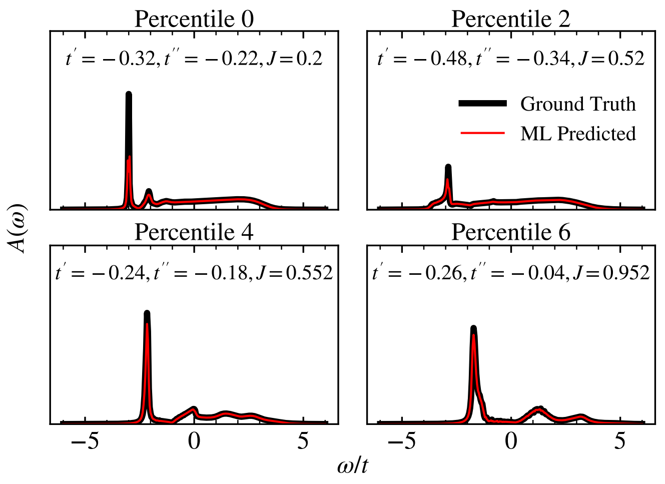


FIG. 5. Comparison of the KNN-predicted DOS and the ground truth (the SCBA-generated DOS) for the worst performing data points in the testing set.

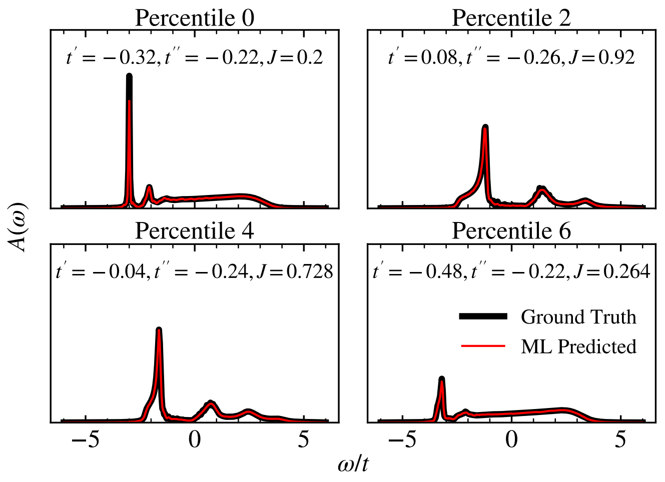


FIG. 6. Comparison of the neural-network-predicted DOS and the ground truth (the SCBA-generated DOS) for the worst performing data points in the testing set.

the examples in the testing set where KNN performs the worst, the KNN prediction is able to reproduce the peak positions and widths almost perfectly, while also performing quite well when reproducing the peak heights.

FFNN.— We then applied a neural network for the same task. We found that with optimized hyperparameters $\mathbf{n} = (3, 170, 340, 510, 680, 850, 1020, 301)$, batch size of 1024, and initial learning rate of 10^{-3} (see Appendix C 1 for details), the neural network with six hidden layers was able to outperform KNN by roughly a factor of 6 in terms of MSE loss. As shown in Fig. 6, even for the worst examples in the testing set, the neural network reproduces peak positions, widths, and heights almost perfectly. For these low-percentile testing examples, the neural network qualitatively appears to reproduce the peak heights better than KNN, which is also manifested quantitatively in its improvement over KNN

TABLE I. Examples of worst percentiles when predicting t', t'', and J with KNN and FFNN, given the DOS. The numbers in the parentheses are ground truth values.

	KNN-Predicted			FFNN-Predicted		
Percentile	$-t'$	t''	J	$-t'$	t''	J
0	0.072(0.02)	0.130(0.14)	0.235(0.232)	0.387(0.38)	0.497(0.50)	0.201(0.200)
1	0.050(0.02)	0.093(0.10)	0.565(0.552)	0.177(0.18)	0.402(0.40)	0.889(0.888)
2	0.235(0.26)	0.140(0.14)	0.657(0.664)	0.022(0.02)	0.201(0.20)	0.266(0.264)
3	0.045(0.02)	0.440(0.44)	0.520(0.520)	0.498(0.50)	0.481(0.48)	0.362(0.360)

in terms of MSE loss.

In addition to excellently reproducing the DOS, the ML algorithms offer a great speedup in computation time over SCBA. While generating the DOS from 130k input combinations using SCBA took over 30 hours,[72] both KNN and the neural network were able to generate the DOS from those same input combinations in seconds: 7.4 seconds for KNN and 1.2 seconds for FFNN; KNN and the neural network saw a 1.5×10^4 and 9×10^4 speedup over SCBA in predicting the DOS, respectively.

Since the forward problem is in principle a matter of low-dimensional data interpolation and a homogeneous grid of values of the parameters can be made in principle for the forward problem, the above studies represent a traditional data interpolation problem in a dense orthorhombic grid of $\{t', t'', J\}$. This explains why *both* KNN and FFNN are good for the forward problem. As for why FFNN outperforms KNN, we note that in KNN, the output spectral intensity at one energy point ω_i is completely independent of the output spectral intensity at another energy point ω_j, i.e., the interpolation was done point-by-point in the ω grid. On the other hand, in FFNN, neurons on the adjacent layers are fully connected, which means correlations among different ω points were utilized in the FFNN algorithm, rendering its better performance.

C. The inverse problem

The inverse problem, which involves predicting the model's parameters from observable quantities, has important experimental implications. Since ARPES experiments produce the spectral function and the DOS, the final goal of inverse modeling would be to predict the Hamiltonian parameters from this available experimental data. In order to make our DOS dataset more experimentally relevant, we shifted every DOS with respect to the top of the quasiparticle valence band [see Fig. 1(b)]. This better mimics experimental data, where absolute energies are not measured, but are instead found relative to the Fermi level [see Fig. 1(a)]. We also limited the dataset to include only examples with $t' < 0$ and $t'' > 0$, i.e, the hole doped case (the case of $t' > 0$ and $t'' > 0$ corresponds to electron doping). As different DOS in our dataset requires shifting of different amounts, this demands an expansion of our energy window. As a result, we now use a 354-point linear grid to sample the shifted DOS. The input is $\mathbf{x}^{(i)} = (A(\omega_1), A(\omega_2), \ldots, A(\omega_{354}))^{(i)}$ and the output is $\mathbf{y}^{(i)} = (t', t'', J)^{(i)}$, where $i = 1, 2, 3, \ldots, \sim 51^3/4$.

KNN.— We first use a KNN to predict the corresponding t', t'', and J, given a DOS. With the trained hyperparameters $k = 9$ and $\alpha = 3$, the results for the worst-percentile predictions are displayed in Table I. We see that for worst percentiles, a KNN is able to predict t'' and J quite accurately, but has trouble with t'. This means that the outstanding problem of predicting t' likely cannot be resolved by this classic modeling method.

FFNN.— We then trained a FFNN for the same task. We found that with the hyperparameters $\mathbf{n} = (301, 256, 128, 64, 32, 3)$, batch size of 128, and a learning rate of 10^{-3} (see Appendix C 2 for details), the neural network is able to significantly outperform KNN, with the MSE being $67\times$ better than that of KNN. As shown in Table I, even for the worst examples in the test set, the neural network can predict at least the first two significant figures. Thus, the neural network offers an accurate approach to prediction of material parameters.

We again highlight that since unlike the forward problem as data interpolation in a low-dimensional homogeneous dense grid, the inverse problem of machine learning spectral functions involves data generalization in a very high-dimensional constrained input space. This key difference explains why while both KNN and FFNN are good for the forward problem, their performances are quite different for the inverse problem.

D. Inverse problem: Finding t

In our above analysis, we have produced and analyzed DOS with t being the energy unit, i.e., $t = 1$. However, ARPES experiments produce DOS that are measured in terms of absolute energy. We thus proceed to analyze the feasibility of obtaining ground truth t (referred to as t_{truth}) from more experimentally realistic DOS. To this end, we examine the following simulation: We start with an SCBA-generated DOS with $t = 1$ from our dataset, shifted with respect to the top of the valence band. We then scale the DOS by using $A(\omega) \to A(\omega\, t_{\text{truth}})/t_{\text{truth}}$, thus producing an "experimental" DOS in units of absolute energy. The task is to find t_{truth} from this "experimental" DOS while pretending that we do not know this t_{truth}.

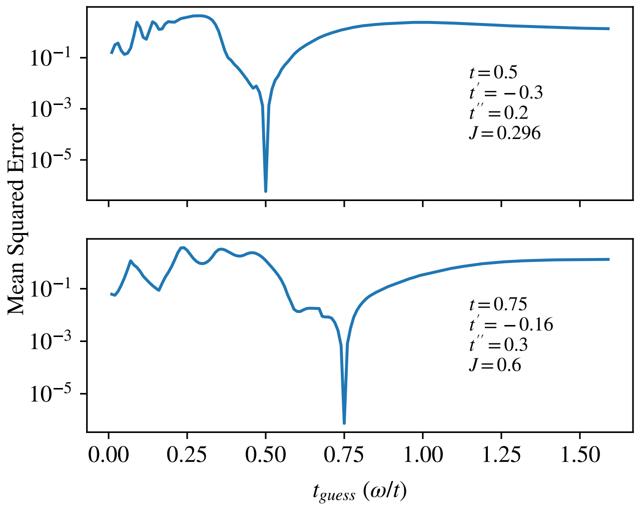


FIG. 7. The mean squared error when rescaling a mock "experimental" DOS in units of $t_{\rm guess}$, and running FFNN to predict the ground truth t', t'', and J.

.

We propose the following algorithm for this task: We add to the ML methods presented in Section III C an outermost loop over various guesses of $t_{\rm truth}$. Specifically, for each $t_{\rm guess}$, we rescale the experimental DOS by using $A(\omega) \to A(\omega/t_{\rm guess})\, t_{\rm guess}$, producing the DOS with $t_{\rm guess}$ being the energy unit; thus, we arrive at the same inverse problem of predicting t', t'', and J with $t = 1$ studied in Section III C. Then, we resample the rescale DOS with the same 354 point energy grid as for the previous inverse problem, using cubic spline interpolation. After that, we run the trained neural network to produce t', t'', and J from the rescaled DOS and calculate the MSE.

The results of this procedure are shown in Fig. 7. We find that this algorithm is able to predict t_{truth} very accurately, as seen by the steep drop in the mean squared error when $t_{\rm guess} = t_{\rm truth}$. Adding such one outermost loop takes advantage of the scalability of the spectral functions [Eq. (6)] and reduces the dimensionality of the model parameter space from 4 to 3, a significant improvement in coping with the curse of dimensionality in big-data ML research.

IV. SUMMARY

We have investigated the potential of ML algorithms for understanding the spectral functions of a hole in the t-t'-t''-J model and found that ML algorithms are well suited for the task. The analysis of the dataset of SCBA-generated spectral functions demonstrates the presence of a continuous mapping between the model parameters and the resulting DOS. Given a set of the model parameters, we found that both KNN and neural networks can produce almost visually identical DOS as SCBA, with a speedup of as much as 9×10^4. We also found that the deep learning neural networks can predict t, t', t'', and J very accurately given a DOS. The difference in the performances of KNN and neural networks for the forward and inverse problems is ultimately due to the principles that the forward problem is a matter of data interpolation in a low-dimensional homogeneous dense grid, whereas the inverse problem involves data generalization in a very high-dimensional constrained input space. With such a speedup in the calculation of DOS, as well as the ability to solve the inverse problem, ML offers a potential tool to search for the model parameters that produce desirable spectral functions. The present method can be directly applied to other cases of energy distribution curves (EDC) such as $A(\mathbf{k}, \omega)$ at constant momentum or the cases of momentum distribution curves (MDC), which are the intensities as a function of momentum at constant energy.[33] Future work will focus on working with experimental data, which are further complicated by instrument resolution and irreducible noise.

DATA AVAILABILITY

The data generated and used in this study are openly available from the Zenodo database.[75]

ACKNOWLEDGMENTS

This work was supported by U.S. Department of Energy (DOE) the Office of Science, Office of Basic Energy Sciences, Materials Sciences and Engineering Division under Contract No. DE-SC0012704. This project was supported in part by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Science Undergraduate Laboratory Internships Program (SULI). This project was supported in part by the Brookhaven National Laboratory (BNL), Condensed Matter Physics and Materials Science Division under the BNL Supplemental Undergraduate Research Program (SURP).

Appendix A: The self-consistent Born approximation

SCBA uses non-crossing Feynman diagrams to calculate Green's function $G(\mathbf{k}, \omega)$ used in Eqs. (2) and (3), which describes the propagation of a particle in the lattice. The self-consistent system of equations to be solved is

$$G(\mathbf{k},\omega) = \left[G^0(\mathbf{k},\omega)^{-1} - \Sigma(\mathbf{k},\omega)\right]^{-1}, \quad \text{(A1)}$$

$$\Sigma(\mathbf{k},\omega) = \sum_{\mathbf{q}} |M(\mathbf{k},\mathbf{q})|^2 G(\mathbf{k}-\mathbf{q},\omega), \quad \text{(A2)}$$

where $\Sigma(\mathbf{k},\omega)$ is the so-called self-energy, $G^0(\mathbf{k},\omega) = \lim_{\eta\to 0^+} \left[\omega + i\eta - \epsilon_{\mathbf{k}}\right]^{-1}$ is the bare Green's function with

$\epsilon_{\mathbf{k}} = 4t' \cos k_x \cos k_y + 2t''[\cos(2k_x) + \cos(2k_y)]$ being the bare dispersion relation of the hole quasiparticle, and $\omega_{\mathbf{q}} = 2J(1-\gamma_{\mathbf{q}}^2)^{1/2}$ is the magnon energy dispersion. The hole-magnon coupling function is $M(\mathbf{k},\mathbf{q}) = 4t(u_{\mathbf{q}}\gamma_{\mathbf{k}-\mathbf{q}} + v_{\mathbf{q}}\gamma_{\mathbf{k}})/\sqrt{N}$ where $u_{\mathbf{q}} = \{[(1-\gamma_{\mathbf{q}}^2)^{-1/2}+1]/2\}^{1/2}$ and $v_{\mathbf{q}} = -\mathrm{sgn}(\gamma_{\mathbf{q}})\{[(1-\gamma_{\mathbf{q}}^2)^{-1/2}-1]/2\}^{1/2}$ with $\gamma_{\mathbf{q}} = [\cos(q_x)+\cos(q_y)]/2$.

In order to generate a high-quality dataset of spectral functions, SCBA samples over a dense mesh for both the hole momentum $\mathbf{k}$ and the magnon momentum $\mathbf{q}$. The sizes of $\mathbf{k}$ and $\mathbf{q}$ can be different while being commensurate, corresponding to the application of twisted boundary conditions.[76] While higher density $\mathbf{k}$ and $\mathbf{q}$ sampling leads to higher quality spectral functions, they are also more computationally expensive. After testing various combinations of $\mathbf{k}$ and $\mathbf{q}$ sampling densities, we found that above sampling densities of a 128×128 lattice for $\mathbf{k}$ and a 32×32 lattice for $\mathbf{q}$, the results converge.

Appendix B: Principal component analysis

Given the training set $\mathbb{T} = \left\{\left(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\right)\right\}$ of size N where $\mathbf{x}^{(i)}$ is an n-dimensional vector and $\mathbf{y}^{(i)}$ is a m-dimensional vector, we begin with the $m \times N$ matrix $Z = \left(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots, \mathbf{y}^{(N)}\right)$ with each column representing a $\mathbf{y}^{(i)}$. The m rows of Z are then each shifted so that the mean of every raw is zero, that is, the center of the data is translated to the origin of the m-dimensional space, which does not change how the data points are positioned relative to each other. The $m \times m$ covariance matrix is given by

$$\mathbf{C_Z} = \frac{1}{m}\mathbf{Z}\mathbf{Z}^{\mathrm{T}} \tag{B1}$$

The principal components of $\mathbf{C_Z}$ are just its normalized eigenvectors $\mathbf{e}_j$ with $j = 1, 2, 3, \ldots, m$, arranged according to their corresponding eigenvalues (variance) in descending order. $\mathbf{e}_1$ is the direction in the m-dimensional space with the largest variance in Z, $\mathbf{e}_2$ is the direction with the second largest variance, and so on. One can use the eigenvalues to determine the proportion of the variation that each principal component accounts for. If $\mathbf{e}_1$ and $\mathbf{e}_2$ account for the vast majority of the variation in the data, a 2D graph, using only $\mathbf{e}_1$ and $\mathbf{e}_2$ as the axes, would be a good approximation of an unimaginable m-dimensional graph. The coordinates of $\mathbf{y}^{(i)}$ projected into the 2D subspace is given by

$$\mathbf{z}^{(i)} \equiv (z_1, z_2)^{(i)} = (\mathbf{e}_1 \cdot \mathbf{y}^{(i)}, \mathbf{e}_2 \cdot \mathbf{y}^{(i)}). \tag{B2}$$

Then, a 2D color map can be produced by coloring all the $\mathbf{z}^{(i)}$ points according to the value of an element in the vector $\mathbf{x}^{(i)}$, so we can obtain n such 2D color maps. Among them, those appearing to be structured and smooth in the color gradients suggesting that the corresponding input parameters can be continuously mapped to spectral functions, making ML algorithms well suited for the task.

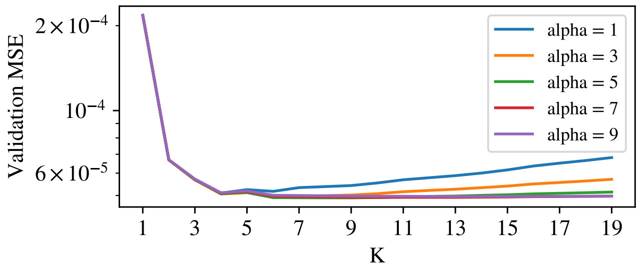


FIG. 8. Validation loss for the forward problem for different values of k, and α.

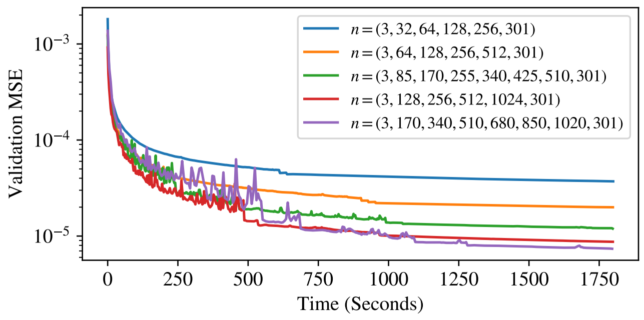


FIG. 9. Validation loss over time for various different FFNN architectures, with $\texttt{bs} = 1024$, and $\texttt{lr} = 10^{-3}$, for the forward problem.

This also suggests that the inverse problem of mapping spectral functions to input parameters is feasible.

Appendix C: Total mean squared error

1. MSE for the forward problem

Hyperparameter tuning was performed by optimizing hyperparameters to reduce validation set error.

For KNN, we tested various values of α and k via grid search. Fig. 8 shows a plot of the validation mean squared error for various values of k, and α, including the optimal value $\alpha = 5$ and $k = 9$.

For FFNN, we tuned hyperparameters with a combination of hand tuning and grid search. The architecture of the neural network $\mathbf{n}$ was of particular interest in hyperparameter tuning. We tested a variety of architectures with different number of layers, which "ramped" up to a different number of neurons in the final hidden layer. Fig. 9 shows a plot of the validation error over time for different architectures over a 30 minute[72] time period after which the validation error changes little.

For the architecture design, we tested various "linear ramps" which simply ramp from 3 input neurons to the 301 output neurons linearly. For example, a linear-ramp architecture with 3 hidden layers would have $\mathbf{n} = (3, 77, 151, 225, 301)$. We found that while these linear ramps intuitively made more sense, they trained

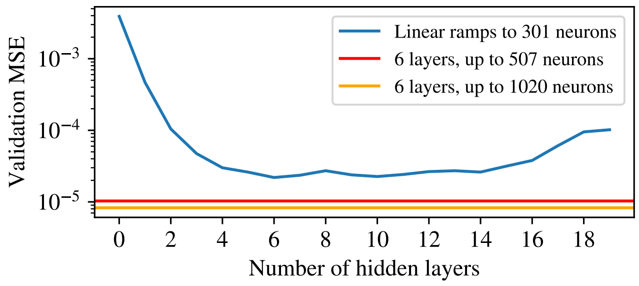


FIG. 10. Validation loss for the forward problem for linear ramps with different number of hidden layers. These are compared to architectures that include more than 301 neurons in the hidden layers.

considerably less efficiently than architectures which had more than 301 neurons in the hidden layers. We see in Fig. 10 that after training for 30 minutes, even the best linear ramps perform worse on the validation set than architectures which include larger hidden layers.

Other hyperparameters include the batch size and the learning rate. The batch size (`bs`) is the size of a subset of $\mathbb{T}$ fed to the neural network to perform a single gradient update. One epoch of training completes after all the training data have been fed through the network (in a randomized order each time). The learning rate (`lr`) is the base step size for tuning weights towards the optimization direction (along gradient descent) and is scheduled to decrease by a factor of 2 when no improvement is realized after 10 epochs. For the forward problem, we find optimized hyperparameters $\mathbf{n} = (3, 170, 340, 510, 680, 850, 1020, 301)$, $\texttt{bs} = 1024$, and $\texttt{lr} = 10^{-3}$.

After optimizing hyperparameters using the validation set for both KNN and FFNN, the following MSE results are derived from the performance of the ML models on the testing set: 4.71×10^{-5} for KNN and 7.24×10^{-6} for FFNN.

2. MSE for the inverse problem

For KNN, we again used grid search to tune hyperparameters.

For FFNN, while we found that an architecture of $\mathbf{n} = (354, 256, 128, 64, 32, 3)$ together with $\texttt{bs} = 128$ and $\texttt{lr} = 10^{-3}$ after hyperparameter tuning, we note that several architectures, which ramped down from 354 to 3 neurons, performed similarly on the validation set.

After optimizing hyperparameters using the validation set for both KNN and FFNN, the following MSE results are derived from the performance of the ML models on the testing set: 4.19×10^{-5} for KNN and 6.29×10^{-7} for FFNN.

* wyin@bnl.gov

[1] E. Dagotto, Rev. Mod. Phys. **66**, 763 (1994).
[2] P. A. Lee, N. Nagaosa, and X.-G. Wen, Rev. Mod. Phys. **78**, 17 (2006).
[3] S. Schmitt-Rink, C. M. Varma, and A. E. Ruckenstein, Phys. Rev. Lett. **60**, 2793 (1988).
[4] P. D. Johnson, G. Xu, and W.-G. Yin, eds., *Iron-Based Superconductivity*, Springer Series in Materials Science, Vol. 211 (Springer International Publishing Switzerland, 2015).
[5] W.-G. Yin, C.-C. Lee, and W. Ku, Phys. Rev. Lett. **105**, 107004 (2010).
[6] Y. Cao, V. Fatemi, S. Fang, K. Watanabe, T. Taniguchi, E. Kaxiras, and P. Jarillo-Herrero, Nature **556**, 43 (2018).
[7] G. Ji, M. Xu, L. H. Kendrick, C. S. Chiu, J. C. Brüggenjürgen, D. Greif, A. Bohrdt, F. Grusdt, E. Demler, M. Lebrat, and M. Greiner, Phys. Rev. X **11**, 021022 (2021).
[8] J. Koepsell, D. Bourgund, P. Sompet, S. Hirthe, A. Bohrdt, Y. Wang, F. Grusdt, E. Demler, G. Salomon, C. Gross, and I. Bloch, Science **374**, 82 (2021).
[9] J. Koepsell, J. Vijayan, P. Sompet, F. Grusdt, T. A. Hilker, E. Demler, G. Salomon, I. Bloch, and C. Gross, Nature **572**, 358 (2019).
[10] A. Bohrdt, C. S. Chiu, G. Ji, M. Xu, D. Greif, M. Greiner, E. Demler, F. Grusdt, and M. Knap, Nature Physics **15**, 921 (2019).
[11] C. S. Chiu, G. Ji, A. Bohrdt, M. Xu, M. Knap, E. Demler, F. Grusdt, M. Greiner, and D. Greif, Science **365**, 251 (2019).
[12] P. T. Brown, D. Mitra, E. Guardado-Sanchez, R. Nourafkan, A. Reymbaut, C.-D. Hébert, S. Bergeron, A.-M. S. Tremblay, J. Kokalj, D. A. Huse, P. Schauß, and W. S. Bakr, Science **363**, 379 (2019).
[13] A. Mazurenko, C. S. Chiu, G. Ji, M. F. Parsons, M. Kanász-Nagy, R. Schmidt, F. Grusdt, E. Demler, D. Greif, and M. Greiner, Nature **545**, 462 (2017).
[14] J. H. Nyhegn, K. K. Nielsen, and G. M. Bruun, Phys. Rev. B **106**, 155160 (2022).
[15] K. K. Nielsen, T. Pohl, and G. M. Bruun, Phys. Rev. Lett. **129**, 246601 (2022).
[16] F. C. Zhang and T. M. Rice, Phys. Rev. B **37**, 3759 (1988).
[17] A. Damascelli, Z. Hussain, and Z.-X. Shen, Rev. Mod. Phys. **75**, 473 (2003).
[18] D. S. Marshall, D. S. Dessau, A. G. Loeser, C.-H. Park, A. Y. Matsuura, J. N. Eckstein, I. Bozovic, P. Fournier, A. Kapitulnik, W. E. Spicer, and Z.-X. Shen, Phys. Rev. Lett. **76**, 4841 (1996).
[19] R. Eder, Y. Ohta, and G. A. Sawatzky, Phys. Rev. B **55**, R3414 (1997).
[20] C. Kim, P. J. White, Z.-X. Shen, T. Tohyama, Y. Shibata, S. Maekawa, B. O. Wells, Y. J. Kim, R. J. Birgeneau, and M. A. Kastner, Phys. Rev. Lett. **80**, 4245 (1998).
[21] W.-G. Yin, C.-D. Gong, and P. W. Leung, Phys. Rev. Lett. **81**, 2534 (1998).
[22] B. O. Wells, Z. X. Shen, A. Matsuura, D. M. King, M. A. Kastner, M. Greven, and R. J. Birgeneau, Phys. Rev. Lett. **74**, 964 (1995).
[23] F. Ronning, C. Kim, K. M. Shen, N. P. Armitage, A. Dam-

ascelli, D. H. Lu, D. L. Feng, Z.-X. Shen, L. L. Miller, Y.-J. Kim, F. Chou, and I. Terasaki, Phys. Rev. B **67**, 035113 (2003).
[24] A. Nazarenko, K. J. E. Vos, S. Haas, E. Dagotto, and R. J. Gooding, Phys. Rev. B **51**, 8676 (1995).
[25] B. Kyung and R. A. Ferrell, Phys. Rev. B **54**, 10125 (1996).
[26] T. Xiang and J. M. Wheatley, Phys. Rev. B **54**, R12653 (1996).
[27] W.-G. Yin and W. Ku, Phys. Rev. B **79**, 214512 (2009).
[28] V. I. Belinicher, A. L. Chernyshev, and V. A. Shubin, Phys. Rev. B **54**, 14914 (1996).
[29] P. W. Leung, B. O. Wells, and R. J. Gooding, Phys. Rev. B **56**, 6320 (1997).
[30] T. K. Lee and C. T. Shih, Phys. Rev. B **55**, 5983 (1997).
[31] E. Pavarini, I. Dasgupta, T. Saha-Dasgupta, O. Jepsen, and O. K. Andersen, Phys. Rev. Lett. **87**, 047003 (2001).
[32] Z.-D. Yu, Y. Zhou, W.-G. Yin, H.-Q. Lin, and C.-D. Gong, Phys. Rev. B **96**, 045110 (2017).
[33] T. Valla, A. V. Fedorov, P. D. Johnson, B. O. Wells, S. L. Hulbert, Q. Li, G. D. Gu, and N. Koshizuka, Science **285**, 2110 (1999).
[34] R. B. Laughlin, Phys. Rev. Lett. **79**, 1726 (1997).
[35] M. R. Carbone, MRS Bulletin **47**, 968–974 (2022).
[36] M. Krenn, R. Pollice, S. Y. Guo, M. Aldeghi, A. Cervera-Lierta, P. Friederich, G. dos Passos Gomes, F. Häse, A. Jinich, A. Nigam, Z. Yao, and A. Aspuru-Guzik, Nature Reviews Physics **4**, 761 (2022).
[37] A. Radovic, M. Williams, D. Rousseau, M. Kagan, D. Bonacorsi, A. Himmel, A. Aurisano, K. Terao, and T. Wongjirad, Nature **560**, 41 (2018).
[38] I. Sadeh, F. B. Abdalla, and O. Lahav, Publications of the Astronomical Society of the Pacific **128**, 104502 (2016).
[39] Y. Hashimoto and C.-H. Liu, Universe **8**, 339 (2022).
[40] N. Schanche, A. C. Cameron, G. Hébrard, L. Nielsen, A. H. M. J. Triaud, J. M. Almenara, K. A. Alsubai, D. R. Anderson, D. J. Armstrong, S. C. C. Barros, F. Bouchy, P. Boumis, D. J. A. Brown, F. Faedi, K. Hay, L. Hebb, F. Kiefer, L. Mancini, P. F. L. Maxted, E. Palle, D. L. Pollacco, D. Queloz, B. Smalley, S. Udry, R. West, and P. J. Wheatley, Monthly Notices of the Royal Astronomical Society **483**, 5534 (2018).
[41] J. Schmidt, M. R. G. Marques, S. Botti, and M. A. L. Marques, npj Computational Materials **5**, 83 (2019).
[42] K. Ryan, J. Lengyel, and M. Shatruk, Journal of the American Chemical Society **140**, 10158 (2018).
[43] X. Zheng, P. Zheng, and R.-Z. Zhang, Chem. Sci. **9**, 8426 (2018).
[44] M. R. Carbone, S. Yoo, M. Topsakal, and D. Lu, Phys. Rev. Mater. **3**, 033604 (2019).
[45] S. B. Torrisi, M. R. Carbone, B. A. Rohr, J. H. Montoya, Y. Ha, J. Yano, S. K. Suram, and L. Hung, npj Comput. Mater. **6**, 1 (2020).
[46] R. Jalem, K. Kanamori, I. Takeuchi, M. Nakayama, H. Yamasaki, and T. Saito, Scientific Reports **8**, 5845 (2018).
[47] M. R. Carbone, M. Topsakal, D. Lu, and S. Yoo, Phys. Rev. Lett. **124**, 156401 (2020).
[48] A. Ghose, M. Segal, F. Meng, Z. Liang, M. S. Hybertsen, X. Qu, E. Stavitski, S. Yoo, D. Lu, and M. R. Carbone, (2022), arXiv:2210.00336.
[49] C. D. Rankine and T. Penfold, J. Chem. Phys. **156**, 164102 (2022).
[50] T. Penfold and C. Rankine, Molecular Physics , e2123406 (2022).
[51] J. Carrasquilla and R. G. Melko, Nature Physics **13**, 431 (2017).
[52] C. Miles, M. R. Carbone, E. J. Sturm, D. Lu, A. Weichselbaum, K. Barros, and R. M. Konik, Phys. Rev. B **104**, 235111 (2021).
[53] J. Liu, Y. Qi, Z. Y. Meng, and L. Fu, Phys. Rev. B **95**, 041101 (2017).
[54] S. Tibaldi, G. Magnifico, D. Vodola, and E. Ercolessi, SciPost Phys. **14**, 005 (2023).
[55] E. J. Sturm, M. R. Carbone, D. Lu, A. Weichselbaum, and R. M. Konik, Phys. Rev. B **103**, 245118 (2021).
[56] L.-F. m. c. Arsenault, A. Lopez-Bezanilla, O. A. von Lilienfeld, and A. J. Millis, Phys. Rev. B **90**, 155136 (2014).
[57] N. Walker, S. Kellar, Y. Zhang, and K.-M. Tam, (2020), arXiv:2008.12331.
[58] F. Marsiglio, A. E. Ruckenstein, S. Schmitt-Rink, and C. M. Varma, Phys. Rev. B **43**, 10882 (1991).
[59] G. Martinez and P. Horsch, Phys. Rev. B **44**, 317 (1991).
[60] Z. Liu and E. Manousakis, Phys. Rev. B **44**, 2414 (1991).
[61] Z. Liu and E. Manousakis, Phys. Rev. B **45**, 2425 (1992).
[62] W.-G. Yin and C.-D. Gong, Phys. Rev. B **56**, 2843 (1997).
[63] W.-G. Yin and C.-D. Gong, Phys. Rev. B **57**, 11743 (1998).
[64] E. Manousakis, Phys. Rev. B **75**, 035106 (2007).
[65] E. Manousakis, Physics Letters A **362**, 86 (2007).
[66] T. Valla, T. E. Kidd, W.-G. Yin, G. D. Gu, P. D. Johnson, Z.-H. Pan, and A. V. Fedorov, Phys. Rev. Lett. **98**, 167003 (2007).
[67] P. W. Leung and R. J. Gooding, Phys. Rev. B **52**, R15711 (1995).
[68] N. G. Diamantis and E. Manousakis, New Journal of Physics **23**, 123005 (2021).
[69] G. Biau and L. Devroye, *Lectures on the Nearest Neighbor Method* (Springer International Publishing, 2015).
[70] D. Shepard, Proceedings of the 1968 23rd ACM National Conference ACM '68, 517–524 (1968).
[71] M. Gardner and S. Dorling, Atmospheric Environment **32**, 2627 (1998).
[72] Computational jobs were run in parallel as 208 processes on 2 nodes with 2 Intel Xeon Gold 6230R CPUs @ 2.10GHz/node, 26 cores/CPU, 2 threads/core, and 192GB memory/node.
[73] D. P. Kingma and J. Ba, arXiv:1412.6980 (2014).
[74] S. Wold, K. Esbensen, and P. Geladi, Chemometrics and Intelligent Laboratory Systems **2**, 37 (1987).
[75] J. Lee, M. R. Carbone, and W. Yin, "Data Repository for: Machine-learning the spectral function of a hole in a quantum antiferromagnet," (2023).
[76] W.-G. Yin and W. Ku, Phys. Rev. B **80**, 180402 (2009).