Supporting Information for: Using machine learning to map simulated noisy and laser-limited multidimensional spectra to molecular electronic couplings

Jonathan D. Schultz*[1], Kelsey A. Parker*[1], Bashir Sbaiti[1,2], and David N. Beratan[1,2,3]

[1]*Department of Chemistry, Duke University, Durham, NC 27708, United States*
[2]*Department of Physics, Duke University, Durham, NC 27708, United States*
[3]*Department of Biochemistry, Duke University, Durham, NC 27710, United States*

*Correspondence to jonathan.schultz@duke.edu, kelsey.parker@duke.edu

S1 Additional simulation details

S1.1 Parametrization of the vibronic dimer Hamiltonian

Table S1 details the parameters we used to construct a set of 1424 unique vibronic dimer Hamiltonians. We formulate the Hilbert space for each system with kets of the form

$$|n_{ea}, n_{v_1a}, n_{v_2a}, n_{eb}, n_{v_1b}, n_{v_2b}\rangle \tag{S1}$$

where n_{ei} is the electronic quantum number for molecule i ($i = a, b$) and $n_{v_k i}$ is the vibrational quantum number for vibrational mode k ($k = 1, 2$ for the 1300 and 200 cm^{-1} modes, respectively) for molecule i. For each independent vibrational mode, we constrained the maximum vibrational quanta in each ket to five.

Table S1: All parameters used to yield the 1424 unique vibronic dimer Hamiltonians. Figure 1 of the main text provides a complementary graphical representation.

Parameter	Units	Values ($N_{systems}$)
ϵ	cm^{-1}	14500 (1424)
J_{Coul}	cm^{-1}	-800 (40), -775 (40), -750 (32), -715 (40), -700 (40), -675 (40), -650 (32), -615 (40), -600 (40), -575 (40), -550 (32), -500 (40), -450 (32), -400 (40), -350 (32), -300 (40), -250 (32), -200 (40), -150 (32), -100 (40), -50 (32), 0 (72), 50 (32), 100 (40), 150 (32), 200 (40), 250 (32), 300 (40), 350 (32), 400 (40), 450 (32), 500 (40), 550 (32), 600 (40), 650 (32), 700 (40), 750 (32), 800 (40)
λ_{1300}	unitless	0.0 (178), 0.1 (178), 0.2 (178), 0.3 (178), 0.4 (178), 0.5 (178), 0.6 (178), 0.7 (178)
λ_{200}	unitless	0.0 (176), 0.1 (312), 0.2 (312), 0.3 (312), 0.4 (312)

S1.2 Simulations of multidimensional spectra

We used the nonlinear response function formalism, in which the nonlinear molecular response function is calculated from a combination of different pathways in Liouville space.[1] The transition dipole operator for a light-matter interaction is written[2] in the Condon approximation as,

$$\mu(\tau_i) = c^\dagger + c \tag{S2}$$

where τ_i reflects the instantaneous time at which the impulsive light-matter interaction occurs.[2,3] Free propagation of the wavefunction under the system Hamiltonian during the time between two light-matter interactions j and k is achieved with the time-evolution operator,

$$U(\Delta t_{jk}) = e^{-iH_{sys}\Delta t_{jk}}. \tag{S3}$$

To lower the computational cost of time-propagation, we partition the transition dipole and time-evolution operators into blocks based on the matrix indices of the electronic manifolds. For a generic operator O, the operator O_{jk} is a subspace of O corresponding to the j and k block indices. This notation ensures that,

$$\mu_{ge} \left|g(\tau_j)\right\rangle = \left|e(\tau_j)\right\rangle \tag{S4}$$
$$U_{ee}(\Delta t_{jk}) \left|e(\tau_j)\right\rangle = \left|e(\tau_k)\right\rangle \tag{S5}$$

where we have represented the S_0 and S_1 states with g and e, respectively.

Forcing τ_j and τ_k to represent sequential moments in time ($j = k+1$), we simplify the notation with $\Delta t_{jk} = t_k$, in turn recovering the common notation of the coherence (t_1), waiting (t_2), and rephasing (t_3) time delays in 2DES. The third-order nonlinear response functions are thus,[3]

$$R_1(t_1,t_2,t_3) = \left\langle i\right| U_{gg}^\dagger(t_1)U_{gg}^\dagger(t_2)U_{gg}^\dagger(t_3)\mu_{eg}^\dagger U_{ee}(t_3)\mu_{ge}U_{gg}(t_2)\mu_{eg}^\dagger U_{ee}(t_1)\mu_{ge}\left|i\right\rangle \tag{S6}$$

$$R_2(t_1,t_2,t_3) = \left\langle i\right| \mu_{ge}^\dagger U_{ee}^\dagger(t_1)U_{ee}^\dagger(t_2)\mu_{eg}U_{gg}^\dagger(t_3)\mu_{eg}^\dagger U_{ee}(t_3)U_{ee}(t_2)\mu_{ge}U_{gg}(t_1)\left|i\right\rangle \tag{S7}$$

$$R_3(t_1,t_2,t_3) = \left\langle i\right| \mu_{ge}U_{ee}^\dagger(t_1)\mu_{eg}U_{gg}^\dagger(t_2)U_{gg}^\dagger(t_3)\mu_{eg}^\dagger U_{ee}(t_3)\mu_{ge}U_{gg}(t_2)U_{gg}(t_1)\left|i\right\rangle \tag{S8}$$

$$R_4(t_1,t_2,t_3) = \left\langle i\right| U_{gg}^\dagger(t_1)\mu_{ge}^\dagger U_{ee}^\dagger(t_2)\mu_{eg}U_{gg}^\dagger(t_3)\mu_{eg}^\dagger U_{ee}(t_3)U_{ee}(t_2)U_{ee}(t_1)\mu_{ge}\left|i\right\rangle \tag{S9}$$

where R_n is the response function for Liouville pathway n, and $|i\rangle$ is the initial state. For simplicity, we assume that the system begins in the global ground state (i.e., $|i\rangle = |0,0,0,0,0,0\rangle$, following the form of eq S1). Eqs S6 and S9 correspond to the non-rephasing ground-state bleach (GSB) and stimulated emission (SE) pathways, respectively, while eqs S7 and S8 are the rephasing GSB and SE pathways, respectively.

We simulated all spectra in the rotating frame[2,4] by removing one or two electronic quanta from the diagonal entries of the blocks corresponding to the singly excited manifold of the electronic Hamiltonian. We included phenomenological effects of system-bath interactions with the lineshape function,

$$g(t) = \Delta E^2 t_c^2 e^{-\frac{t_i}{t_c}+(\frac{t_i}{t_c}-1)} \tag{S10}$$

where ΔE captures energy gap fluctuations with correlation time t_c.[5] We incorporated the lineshape function $g(t)$ by multiplying $R_{i=1,2,3,4}(t_1,t_2,t_3)$ along each time dimension with $e^{-g(t)}$. Following fast Fourier transformations along t_1 and t_3, we calculated absorptive 2DES spectra with,

$$R_{Abs}(\omega_1,t_2,\omega_3) = \sum_{i=1}^{4} \mathrm{Re}[R_i(\omega_1,t_2,\omega_3)]. \tag{S11}$$

For all parameters of the simulations, including those of the finite lineshapes, we defined values (Table S2) to reflect typical conditions of 2DES experiments (see Refs. 2, 6, and 7, for example).

Table S2: All parameters used to generate 2DES spectra from the 1424 vibronic dimer Hamiltonians.

Parameter	Units	Values
t_1	fs	$[0:3:186]^{a}$
t_2	fs	$[0:5:1245]^{a}$
t_3	fs	$[0:3:186]^{a}$
n_{pad}	unitless	256^{b}
ΔE	cm^{-1}	1300 $(t_1, t_3)^{c}$ 125 (t_2)
t_c	fs	40 $(t_1, t_3)^{c}$ 300 (t_2)

[a] Format: [minimum value: step size: maximum value].

[b] Length of zero padding prior to FFT operations.

[c] Optical coherences during t_1 and t_3 dephase faster than coherences during t_2. Hence, we scaled the lineshape parameters accordingly.

S1.3 Automated image resizing

The simulations described in the previous section yield spectra of size 256×256 along the $\omega_1 \times \omega_3$ dimensions (corresponding to approximately 11075 cm^{-1} along each frequency axis). Due to the energy scales of the system Hamiltonians and the optical response simulation parameters, there is inherently low signal and therefore low information in the outer regions of the spectra (e.g., Figure S1a). To generate smaller, more computationally tractable spectra inputs to the NN, and to remove spectral regions with low information, we used an in-house script to automatically trim the spectra around a central coordinate $(\omega_{1c}, \omega_{3c})$. The algorithm produces "resized" spectra with size 151×151 (approximately 6500 cm^{-1} along each frequency axis; see Figure S1b), as mentioned in the main text.

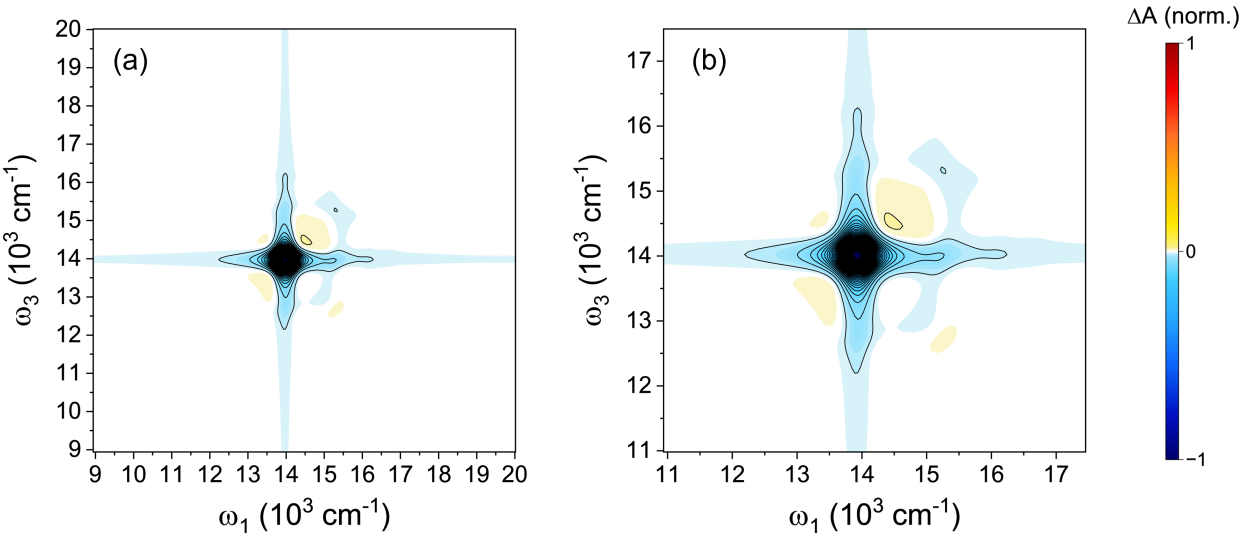

Figure S1: Example spectra (a) before and (b) after the automated trimming and centering algorithm. The example spectra correspond to a system Hamiltonian with parameter values: $J_{Coul} = -500$ cm^{-1}, $\lambda_{1300} = 0.2$, $\lambda_{200} = 0$.

The trimming method works by first determining a central coordinate where the signal is most concentrated $(\omega_{1c}, \omega_{3c})$. The trimmed spectra are then generated by collecting the desired sized subset of data around the central coordinate. It is possible for the new spectra generated by our resizing algorithm to include indices outside the $\omega_1 \times \omega_3$ bounds of the original spectra; this would be the case, for example, if the signals in a given spectrum were highly concentrated in a corner with very low signal everywhere else. In this case, the new pixels included in the spectrum would have zero signal. We did not note any cases where this occurred in our data sets as our spectra

did not have this type of signal distribution.

S2 Additional machine learning details

S2.1 Architecture

As described in the main text, we extended the ML workflows of Parker and coworkers.[8] Our ML framework relies on the PyTorch library[9] in Python. We herein use PyTorch notation to describe the relevant functions, sub-libraries, etc. in our ML methods. We employed the Adam optimizer (torch.optim.Adam) to minimize the CrossEntropyLoss cost function (torch.nn.CrossEntropyLoss). We identified the model predictions as the class index with the highest score (torch.max), and computed probabilities with the Softmax function (torch.nn.Softmax) along the class dimension. Prior to conducing performance analyses (e.g., F1 scores and top-k accuracies), we converted the probabilities output from torch.nn.Softmax() to a NumPy array.

S2.2 Reproducibility

We used seeds in this work to enable reproducibility between runs. We initialized the trainable parameters of the NN consistently by setting the PyTorch seed (torch.manual_seed) in all runs to 2942. Prior to splitting the dataset in to training and testing subsets, we shuffled the dataset with NumPy's random number generation (RNG) sub-library, which we seeded with 72067. Thus, all ML trials used the same subsets of the dataset for training and testing. We also seeded the NumPy RNG for generating Hamiltonian-specific noise profiles while maintaining the Hamiltonian-noise profile correspondence between different ML trials (see Section S2.4 for further details).

S2.3 Hyperparameter optimization

We performed a grid-search to determine optimal values for the hidden layer size, learning rate, and dropout hyperparameters. Table S3 provides all the values of the grid search and Figure S2 shows model performance for each combination. We found that the parameter set, abbreviated as [hidden layer size, learning rate, dropout], of [500, 0.001, 0.2] yielded the maximum F1 score. This set differs from the parameter set [300, 0.001, 0.2] that we used for this study due to our choice to trade a minor performance loss for a smaller hidden layer (F1 = 0.8448 and 0.8457 for 300 and 500 neurons, respectively), in turn enabling faster training times. For the number of epochs, we examined the behavior of the loss function versus epoch and determined 30 epochs to be a sufficient compromise between high testing accuracy and tractable training times (Figure S5). To avoid the massive computational cost required to optimize hyperparameters for each uniquely polluted dataset, we performed the grid search solely on the clean dataset and kept the resulting hyperparameters constant for all other ML trials.

Table S3: Parameters of hyperparameter grid search

Parameter	Values
Hidden layer size	100, 150, 200, 250, 300[a], 350, 400, 450, 500
Learning rate	0.01, 0.0075, 0.005, 0.0025, 0.001[a]
Dropout	0.2[a], 0.4

[a] Hyperparameters used for all trials in the manuscript.

We also used the Optuna optimization package[10] to explore the effect of additional hidden layers. Note that we performed this analysis on a version of the dataset with a lower numerical precision data type (float32, as opposed to the float64 data type used in the rest of our study). From an Optuna scan with 123 trials (105 pruned, 18 complete), we found marginal performance gains from the addition of a second hidden layer (see Table S4 for example trials).

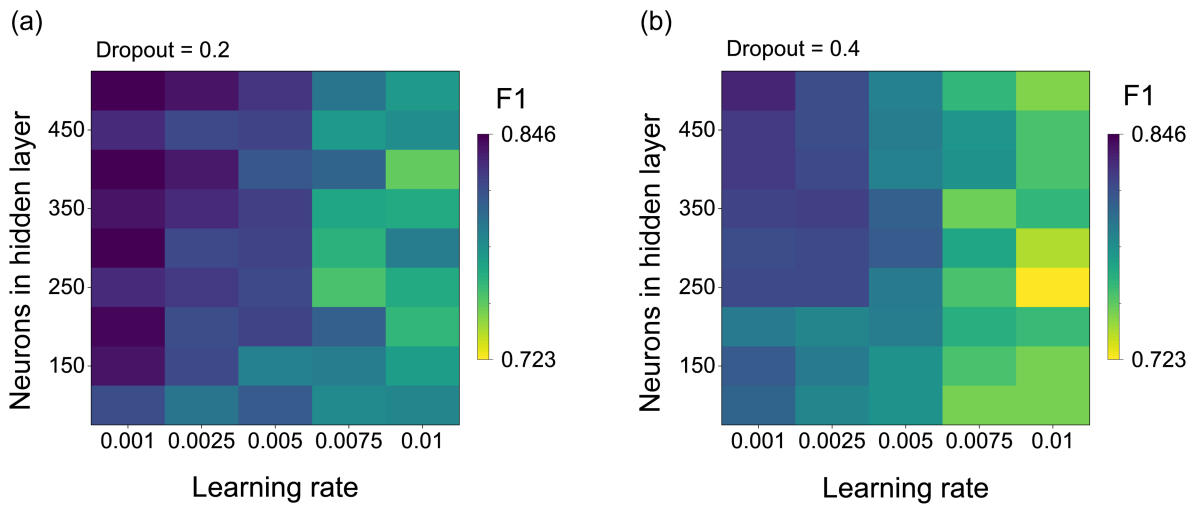

Figure S2: Test F1 scores (macro-averaged) as a function of hidden layer size and learning rate with dropout equal to (a) 0.2 and (b) 0.4.

Table S4: Hyperparameter search with Optuna.

Parameter	Example trial 1	Example trial 2
Number of hidden layers	1	2
Hidden layer size (layer 1)	344	276
Hidden layer size (layer 2)	–	434
Learning rate	0.0067	0.0014
Dropout (layer 1)	0.1012	0.0307
Dropout (layer 2)	–	0.0141
Optuna performance	0.556	0.593

S2.4 Random noise injection

We used the normal distribution function of Numpy's Random sub-library to generate Gaussian noise profiles along each signal dimension, with the sole exception that correlated additive noise was invariant along ω_3 (constant baseline offset for each pump slice, as shown in Figure 3b). To most transparently study how noise in the dataset influenced ML, we ensured the following properties of the noise injection: (i) The noise profile injected to a given spectrum in one ML trial was identical to that injected into the same spectrum during a different, independent ML trial; (ii) No two 2DES spectra received identical noise profiles. While looping over the Hamiltonian index to add noise to each spectra, we used the following procedural steps:

1. Call numpy.random.default_rng() with a known seed determined from the system index.
2. Generate a 3D NumPy array of random noise, of size $\omega_1 \times \omega_3 \times t_2$, such that the 2D noise profiles ($\omega_1 \times \omega_3$) are independent between each of the 250 t_2 time points. In the case of correlated additive noise, we generated a 2D NumPy array ($\omega_1 \times t_2$) that was then broadcasted along the ω_3 dimension.

Seeding the NumPy RNG with Hamiltonian-specific seeds ensured property (i), and generating the noise arrays with a single instance of NumPy RNG is highly likely to have ensured (ii). The only variable between ML trials with differing amounts of noise injected was thus the standard deviation of the normal distribution (σ). Note that, as we discussed in the main text, our approach does not treat any noise sources that maintain correlations along t_2.

S2.5 Signal-to-noise ratio

The signal-to-noise ratio (SNR) is a common metric used in experimental spectroscopy. We define the SNR as

$$\mathrm{SNR} = \frac{\sigma_S}{\sigma_N + \alpha} \tag{S12}$$

where σ_S and σ_N are the signal- and noise-widths, respectively, and α is a constant (10^{-10}) to avoid division by zero. We determine the values of σ_S and σ_N as the mean of the absolute-valued-array for each clean spectrum and its corresponding noise profile, respectfully. Figure S3a shows how the SNR depends on the category of noise.

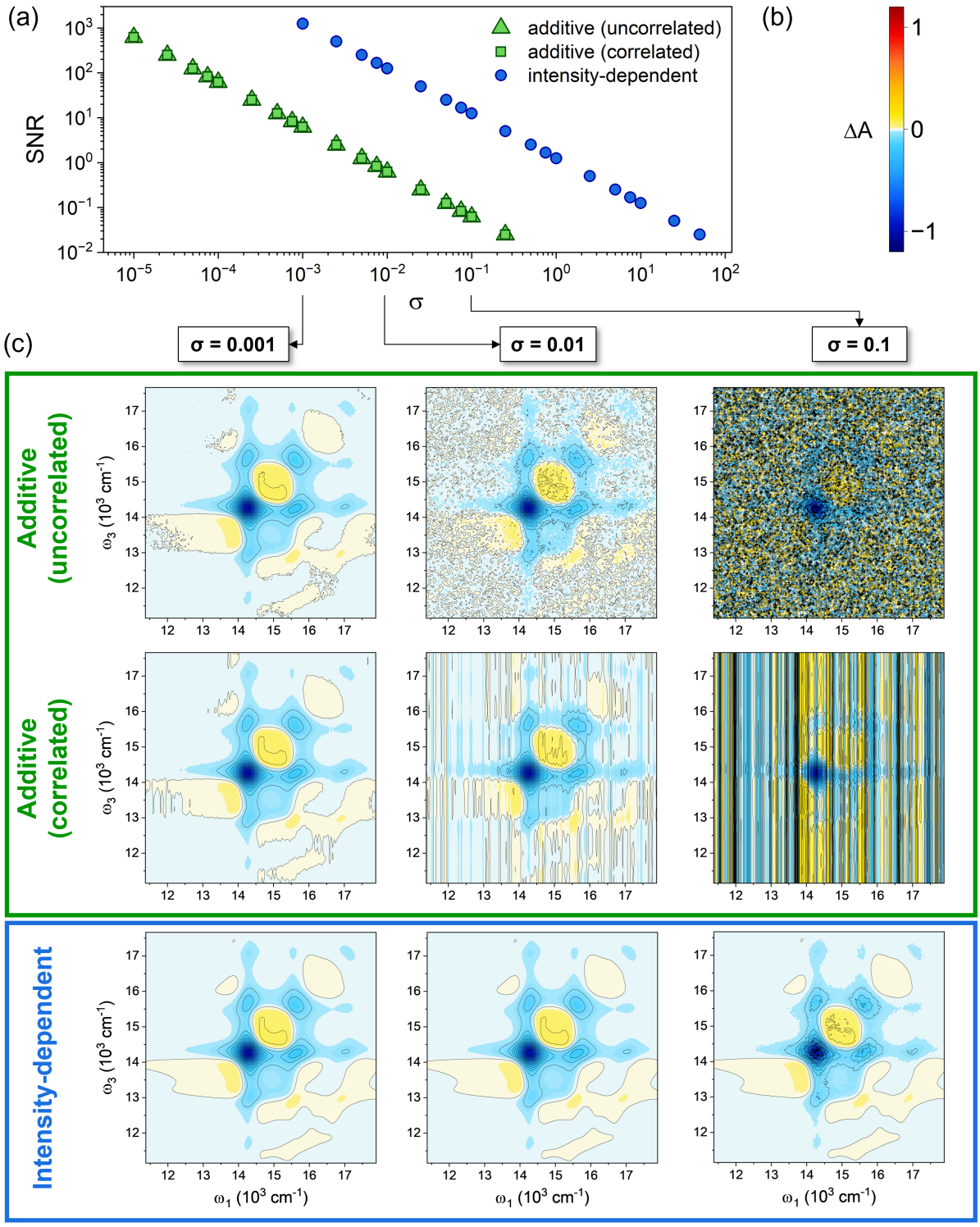

Figure S3: (a) Values of the SNR (averaged over all spectra in the dataset) versus σ for each noise category. (b) Scale bar for the 2D spectra in panel (c). (c) Representative spectra with additive (upper) and intensity-dependent (lower) noise for three σ values.

In practice, experimentalists do not attempt to interpret spectra that are saturated with noise beyond recognition. A similar practice should be incorporated into the training of NNs on noisy spectra. Thus, we defined a threshold SNR (0.01) such that spectra with SNR values below 0.01 were removed from the training and testing datasets. Figure S4 shows how the number of spectra removed from the full dataset as a function of σ. This threshold yielded no dropped spectra for trials with intensity-dependent noise.

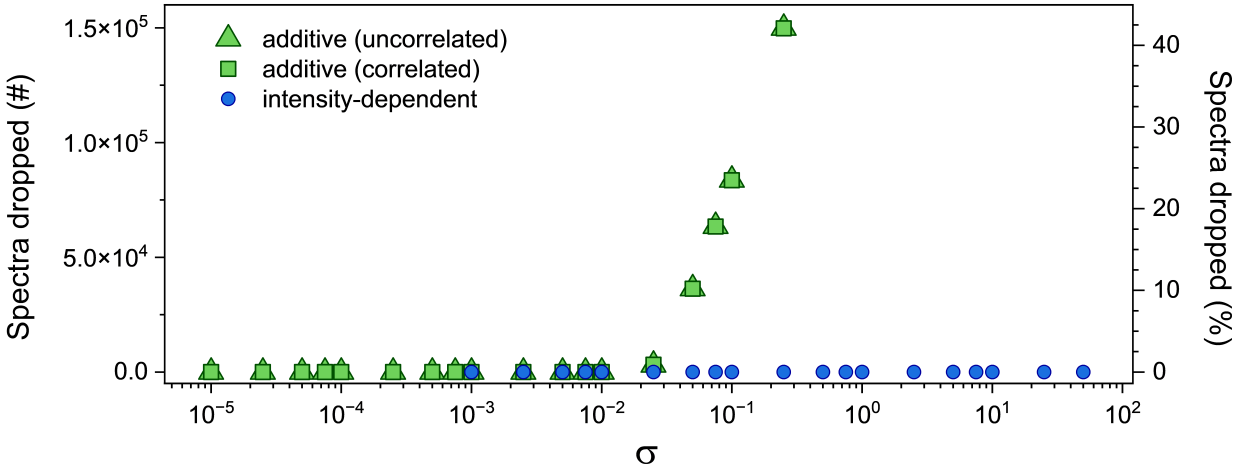

Figure S4: Number of spectra removed from the dataset versus σ for each noise category.

S2.6 Modeling the pump spectrum

We modeled pump spectra in this work with a Gaussian profile,

$$\tilde{E}(\omega) = e^{\frac{-4log(2)(\omega-\omega_c)^2}{\Delta\omega^2}}, \tag{S13}$$

where $\tilde{E}(\omega)$ is the electric field as a function of the pump frequency ($\omega = \omega_1$), ω_c is the carrier frequency, and $\Delta\omega$ is the pulse bandwidth. Since convolution in the time-domain is equivalent to multiplication in the frequency domain,[4] we accounted for effects of the pump pulse spectrum by multiplying $R_{Abs}(\omega_1, t_2, \omega_3)$ by $\tilde{E}(\omega_1)$.

S3 Example spectra

The supplementary documents for this study include several .mp4 video files that show example 2DES data (see Table S5 for file details).

Table S5: Explanations of supplementary video files. All spectra correspond to a Hamiltonian with $\epsilon = 14500$ cm^{-1}, $J_{Coul} = -250$ cm^{-1}, $\lambda_{1300} = 0.6$, and $\lambda_{200} = 0.2$.

File name	Details
clean.mp4	un-polluted spectra (i.e., no effects of noise or pump resonance)
additive_noise_uncorrelated_0.001.mp4	spectra with uncorrelated additive noise ($\sigma_{add} = 0.001$)
additive_noise_uncorrelated_0.01.mp4	spectra with uncorrelated additive noise ($\sigma_{add} = 0.01$)
additive_noise_uncorrelated_0.1.mp4	spectra with uncorrelated additive noise ($\sigma_{add} = 0.1$)[a]
additive_noise_correlated_0.001.mp4	spectra with correlated additive noise ($\sigma_{add} = 0.001$)
additive_noise_correlated_0.01.mp4	spectra with correlated additive noise ($\sigma_{add} = 0.01$)
additive_noise_correlated_0.1.mp4	spectra with correlated additive noise ($\sigma_{add} = 0.1$)[a]
intensity_noise_0.1.mp4	spectra with intensity-dependent noise ($\sigma_{int} = 0.1$)
intensity_noise_1.mp4	spectra with intensity-dependent noise ($\sigma_{int} = 1$)

[a] Some frames dropped due to low SNR (see Figure S4).

S4 Additional results

Figures S5 through S7 show analysis performed during the model training stage. For clean training and test datasets, the model performance exhibits the expected growth vs. epoch with an eventual plateau (Figure S6). In contrast, in the the ML trial with $\sigma_{add} = 0.25$, the model performance exhibits clear signs of the model memorizing the noise signatures (Figure S7). Specifically, the model performance on the training dataset (Figure S7a) grows rapidly over the first five epochs while the test F1 score (Figure S7b) remains essentially invariant. Thus, all performance gains by the model on the training dataset vs. epoch are associated with memorizing the noise (as opposed to learning transferrable knowledge for the inverse classification problem at hand).

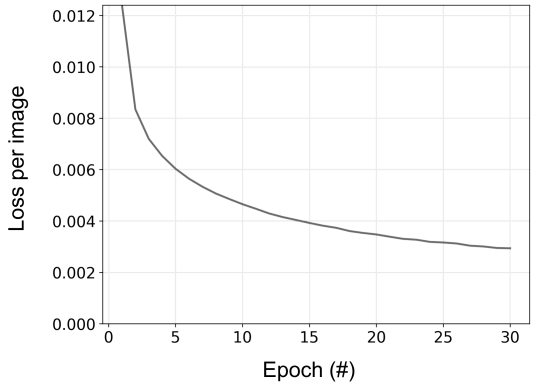

Figure S5: Cross-entropy loss function for the un-polluted dataset.

Figure S8 shows several model performance metrics on the training and test datasets as a function of additive noise σ_{add}. For all datasets and pollutant types, we generally observed similar values for the accuracy and F1 scores (note line overlaps in Figure S8b).

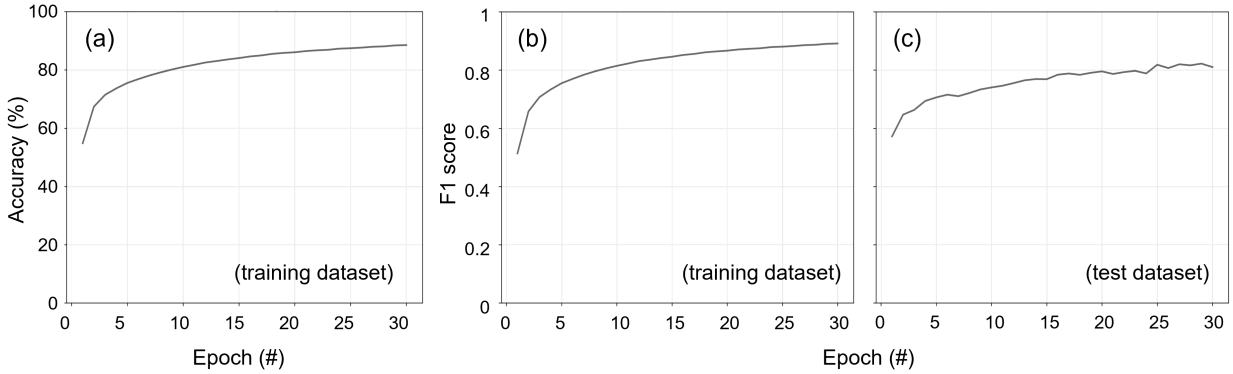

Figure S6: Model performance vs. training epoch calculated as (a) accuracy and (b) F1 score (macro-averaged) on the clean (unpolluted) training dataset, and (c) F1 score (macro-averaged) on the clean (unpolluted) test dataset.

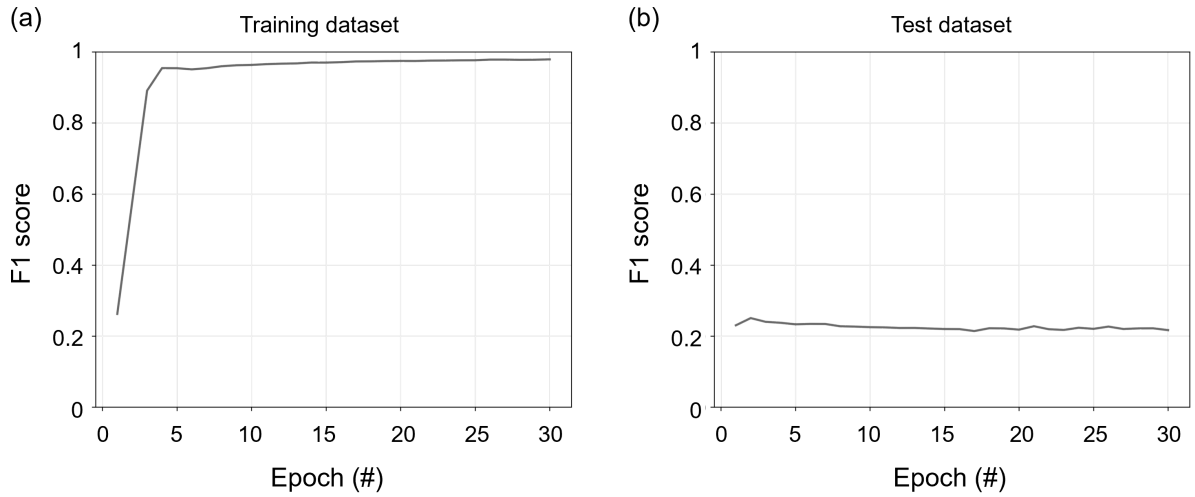

Figure S7: Model performance (macro-averaged F1 score) vs. training epoch calculated on (a) training and (b) test datasets polluted with additive noise ($\sigma_{add} = 0.25$).

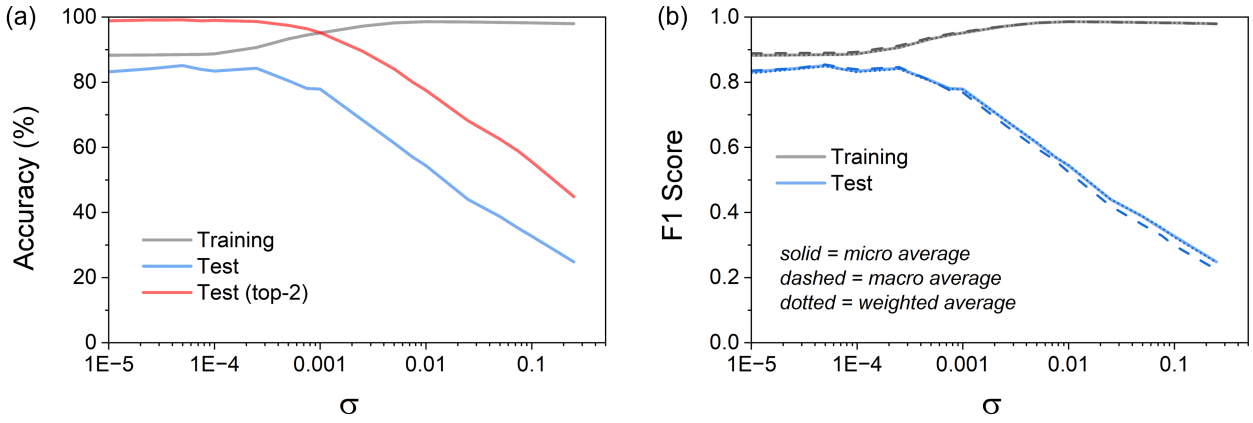

Figure S8: Model (a) accuracy and (b) F1 score metrics for datasets with additive noise. In (b), the micro-, macro-, and weighted-averaged F1 scores are indicated by solid, dashed, and dotted lines, respectively.

Figure S9 shows confusion matrices for ML trials as the amount of each category of noise is varied in the datasets. We find that, in general, misclassifications that are more than one category away from the ground truth become increasingly common as σ increases. For trials with the maximum amounts of uncorrelated additive and intensity-dependent noise (the right-most panels of Figure S9a and Figure S9c, respectively), the misclassifications are particularly prevalent near the center of the confusion matrix. This result suggests that, for these categories of noise, the NN struggles most with dimers that have weak-to-intermediate electronic coupling ($-500 \lessapprox J_{Coul} \lessapprox 500$ cm^{-1}) when the amount of noise in the data is high. The right-most panel of Figure S9b shows similar behavior for the case of maximum correlated additive noise, with the key distinction that the NN disproportionately predicts $J_{Coul} = 0$ cm^{-1}when the ground truth is $0 < |J_{Coul}| < 400$ cm^{-1}.

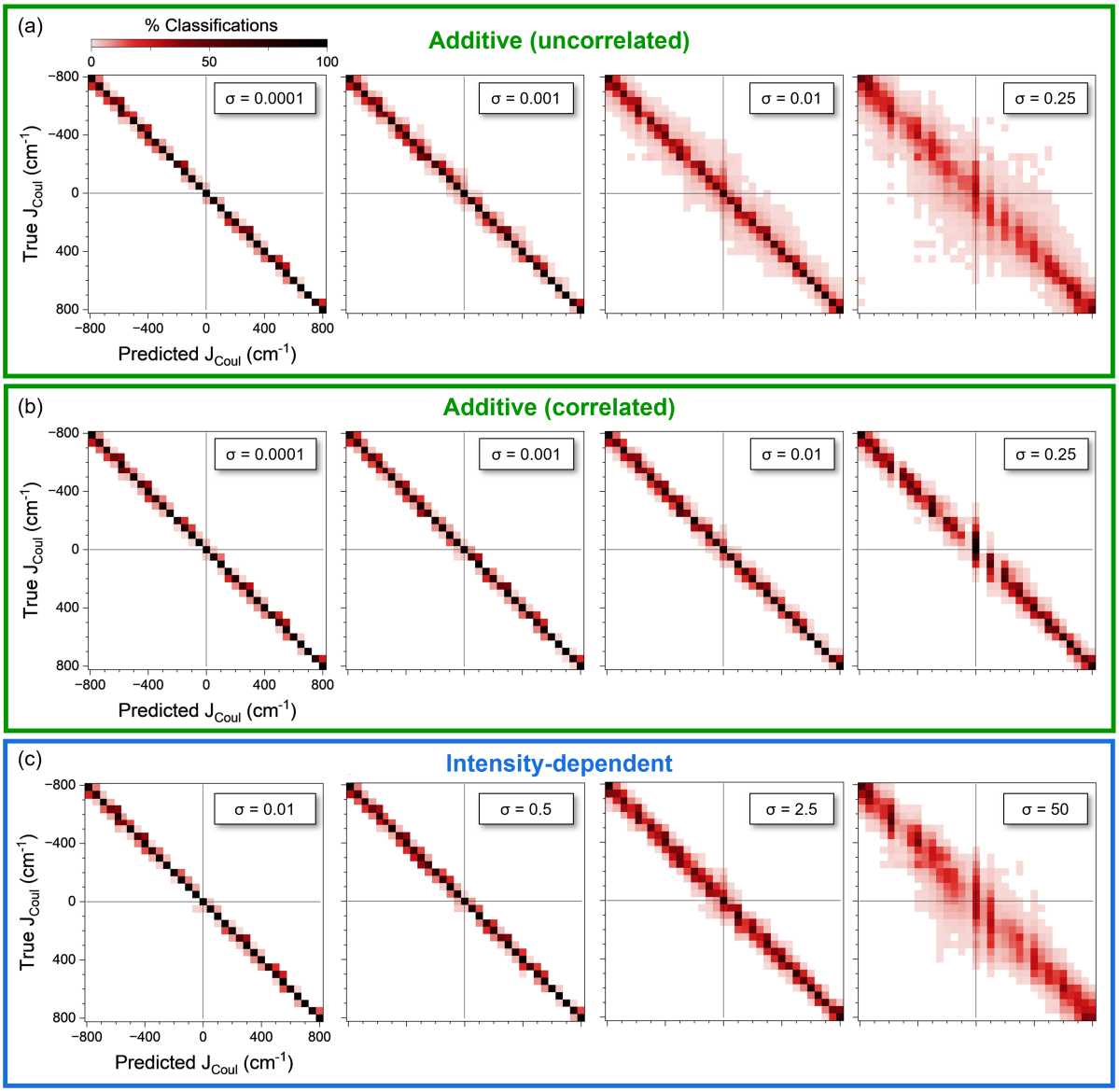

Figure S9: Confusion matrices for datasets polluted with (a) uncorrelated additive, (b) correlated additive, or (c) intensity-dependent noise. Each confusion matrix corresponds to a full ML trial (Figure 2), where the corresponding value of σ is inset.

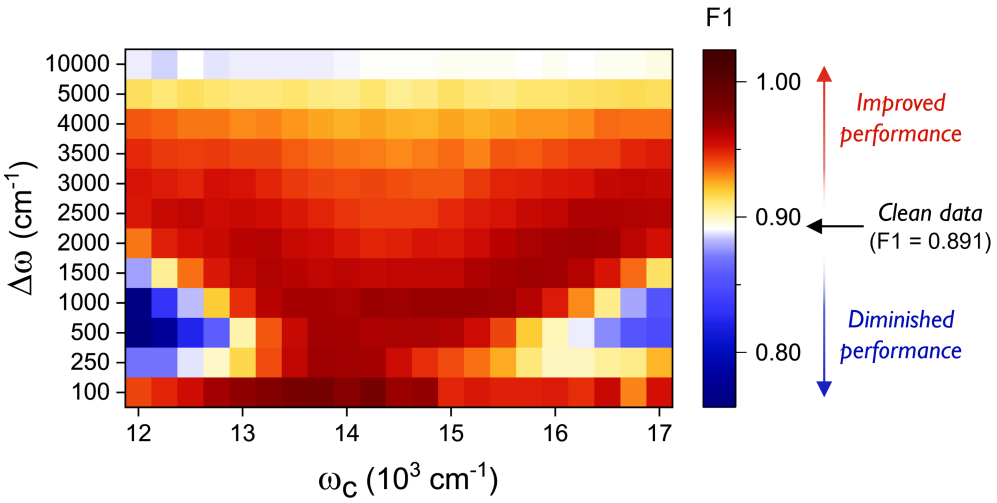

Figure S10: Model F1 score for the training dataset as a function of $\Delta\omega$ and ω_c of the pump pulses. The color scale is based on the F1 score of 0.89129 from the clean dataset.

References

[1] Mukamel S. Principles of Nonlinear Optical Spectroscopy. Oxford series in optical and imaging sciences. Oxford University Press; 1995.

[2] Halpin A, Johnson PJ, Tempelaar R, Murphy RS, Knoester J, Jansen TL, et al. Two-dimensional Spectroscopy of a Molecular Dimer Unveils the Effects of Vibronic Coupling on Exciton Coherences. Nat Chem. 2014;6(3):196-201.

[3] Jansen TlC, Knoester J. Nonadiabatic effects in the two-dimensional infrared spectra of peptides: application to alanine dipeptide. J Phys Chem B. 2006;110(45):22910-6.

[4] Hamm P, Zanni M. Concepts and methods of 2D infrared spectroscopy. Cambridge University Press; 2011.

[5] Kubo R. A Stochastic Theory of Line Shape and Relaxation. In: Ter Haar D, editor. Fluctuation, Relaxation, and Resonance in Magnetic Systems. Oliver & Boyd; 1962. .

[6] Schultz JD, Kim T, O'Connor JP, Young RM, Wasielewski MR. Coupling between Harmonic Vibrations Influences Quantum Beating Signatures in Two-Dimensional Electronic Spectra. J Phys Chem C. 2022;126(1):120-31.

[7] Thyrhaug E, Tempelaar R, Alcocer MJP, Zidek K, Bina D, Knoester J, et al. Identification and characterization of diverse coherences in the Fenna-Matthews-Olson complex. Nat Chem. 2018;10(7):780-6.

[8] Parker KA, Schultz JD, Singh N, Wasielewski MR, Beratan DN. Mapping Simulated Two-Dimensional Spectra to Molecular Models Using Machine Learning. J Phys Chem Lett. 2022;13(32):7454-61.

[9] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. Pytorch: an Imperative Style, High-performance Deep Learning Library. In: Wallach H, Larochelle H, Beygelzimer A, Alché-Buc Fd, Fox E, Garnett R, editors. Advances in Neural Information Processing Systems. vol. 32. Curran Associates, Inc.; 2019. p. 8026-37.

[10] Akiba T, Sano S, Yanase T, Ohta T, Koyama M. Optuna: A Next-generation Hyperparameter Optimization Framework. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2019. .