Supporting Information

Learning A Local-Variable Model of Aromatic and Conjugated Systems

Matthew K. Matlock, Na Le Dang, and S. Joshua Swamidass*

Washington University in St. Louis, School of Medicine, Department of Pathology and Immunology, Saint Louis, MO, USA

E-mail: swamidass@wustl.edu

List of Figures

List of Tables

Supplementary Text

Robustness with Reduced Training Set Size

WAVE models also perform better than other methods with a reduced number of training examples. With just 1,948 training molecules, the N-WAVE model still outperforms weave (conjugated system size RMSE of 18.2 and 12.0, respectively). With scarce data, WAVE model generalization accuracy can be improved by randomizing the choice of starting position in the molecule during each optimization step (Figure S6A). Using this strategy, accuracy improved for both aromatic system size (RMSE 3.82 vs. 3.66; Figure S6B and C) and conjugated system size (RMSE 12.0 vs. 8.92; Figure S6D and E).

Complexity Analysis of the WAVE Architecture

In this analysis, consider a WAVE model where only a fixed number of forward-backward passes are required to apply a trained model to a system of arbitrary size. In the simplest case, on a serial central processing unit (CPU), the time complexity is $O(N)$. We assume that computations are executed in parallel whenever possible on a graphics processing unit (GPU) in constant time. The time complexity of a parallel WAVE is the depth of the breadth-first search, because all computations at the same depth can be computed in parallel. In a linear-1D molecular system with N atoms, there is no branching and no opportunities for parallelism. This yields a linear time complexity $O(N)$ and linear memory complexity $O(N)$. Most organic molecules are approximately planar. In a planar-2D molecular system, the depth of breadth-first search grows in proportion to the radius of the system, with branching and parallelism arising proportionally with N. This yields a sub-linear time complexity $O(\sqrt{N})$ and linear memory complexity $O(N)$. Condensed matter systems are space filling. In a 3D system, branching and parallelism arises proportionally with the volume of the system divided by its width, or proportionally with N. This admits a time complexity of $O(\sqrt[3]{N})$ and maintains linear memory complexity $O(N)$. Moreover, in all cases (1D, 2D,

and 3D), a well-chosen starting atom at the center of the system halves the number of steps required to apply the model.

Complexity Analysis of Weave Architecture

In practice, weave models are designed a fixed number of layers, and only propagate information locally. In the simplest case, on a serial CPU, the time complexity is $O(NL)$ on a system with N atoms. However, parallelism is high because each local aggregation is independent of all others. This yields a favorable complexity of $O(L)$ on a GPU, which is constant no matter how large the system. However, weave models do not propagate information across large systems, which limit their applicability on problems where electrons are widely delocalized.

Complexity Analysis of a Hypothetical Adaptable-Weave Architecture

To enable more efficient long-range information propagation, it is possible that a weave-like system might be devised that can adapt the number of layers to the size of system being modeled. In this way, weave might propagate information across large systems. This model, however, has not been proposed elsewhere and it is beyond our scope to test it here. The number of layers required to propagate information across a system would be proportional to the number of atoms in the system, no matter its dimension. Like the WAVE, therefore, the 1D, 2D, and 3D systems would have time complexity of $O(N)$, $O(\sqrt{N})$, and $O(\sqrt[3]{N})$, respectively, in parallel execution on a GPU. On a serial CPU, however, the time complexity is $O(N^2)$. Although time complexity is comparable to WAVE on a GPU, an adaptable weave architecture would still be substantially less efficient in serial computation.

Complexity Analysis of DAG Architecture

In the literature, a directed-acyclic graph (DAG) architecture has been proposed to model chemicals with deep learning.[1] Though superficially like the WAVE, a DAG network is very different; (1) it computes a different DAG for every atom in the molecule, instead of using just one for the whole molecule, (2) it does not reuse any information between computations, (3) it does not handle bonds that form cycles, and requires a precomputation to collapse rings into pseudorings, and (4) it only makes one pass (equivalent to a WAVE backward pass) through each DAG. Its complexity on a serial system is quadratic $O(N^2)$. On a parallel GPU, it is proportion to the number of atoms times the length of the system; for 1D, 2D, and 3D systems, its time complexity is $O(N^2)$, $O(N^{3/2})$, and $O(N^{4/3})$, respectively. Although this architecture can propagate information across a whole molecule, it cannot perceive rings and it is computationally inefficient. Consequently, it has been excluded from direct testing in this study.

Complexity Analysis of a Hypothetical Restricted Boltzmann Machine

Restricted Boltzmann Machines (RBM) have accuracy comparable to analytic methods in representing quantum mechanics on 2D and 1D lattices with periodic boundary conditions.[2] It is not clear how this method could be scaled up to handle complex 3D quantum systems like organic molecules or condensed phase simulations with several moving and interacting atoms. If an RBM could work, universal approximation theorems would guarantee it could represent quantum mechanics if appropriate trained. However, it would require a large matrix multiply of matrices with dimensions at least as large as the number of atoms N. This matrix multiply has a naïve complexity of $O(N^3)$, but could be improved no better than $O(N^2)$. However, there is no guarantee that even more dimensions would not be required, which would further increase the complexity. Setting aside the difficulty in developing this

approach so it might be applicable to complex systems, this time complexity is substantially higher than others.

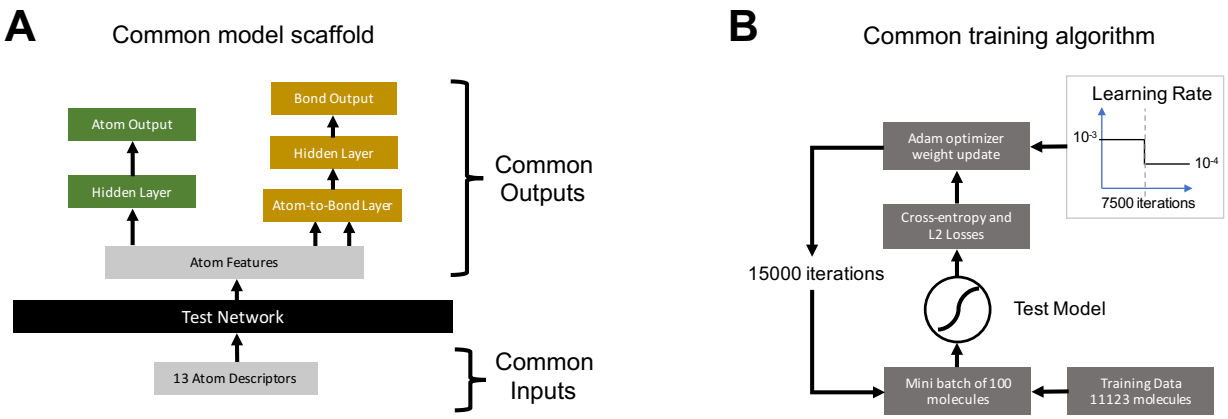

Figure S1: Common input, output and training architectures used for all models in this study. Holding these elements fixed controls experiments so any performance differences arise from differences in the test model's ability to propagate information effectively. **(A)** The same atom descriptors are used as input to all the models. Each model outputs a vector of atom features for each atom, which are then used to compute atom and bond properties via a common output architecture. The atom output (green) is computed by a neural network with one hidden layer. The bond output (gold) combines information from two atoms. These bond features are then passed through a neural network with one hidden layer. All activations in the hidden layers are rectified linear. The binary outputs are sigmoid activations, and the integer outputs are rectified linear activations. **(B)** All models were trained using 15,000 iterations of mini-batch gradient descent with the Adam optimizer, a batch size of 100 molecules, a learning rate of 10^{-3} with a decay to 10^{-4} at 7,500 iterations, cross-entropy loss for binary targets and normalized L2 loss for integer targets.

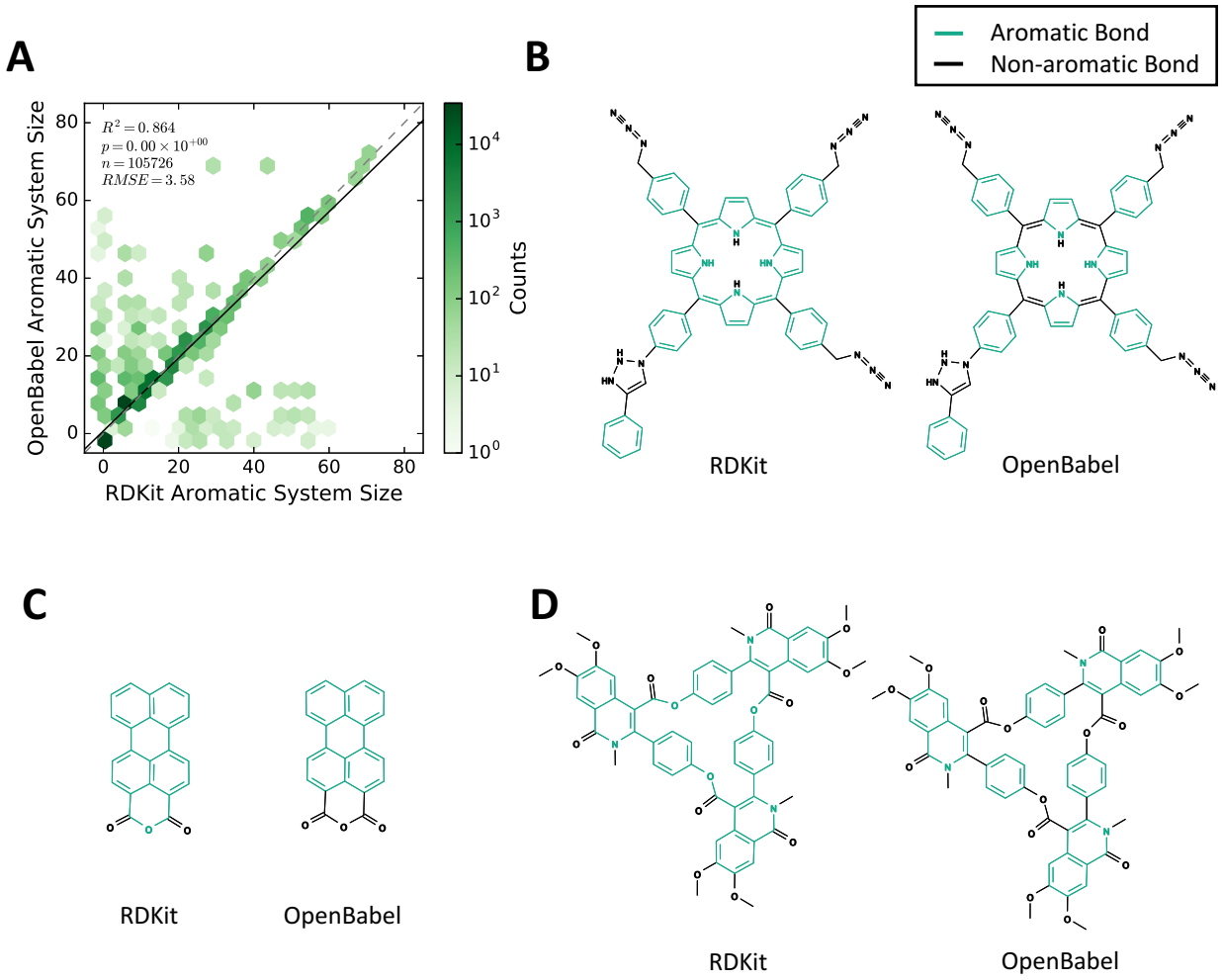

Figure S2: Definitions of aromatic systems vary between detection algorithms. **(A)** Labels of aromatic system size are highly concordant between aromaticity algorithms implemented by two commonly used chemical informatics software packages: RDKit and OpenBabel. However, many exceptions are present in the data. **(B)** RDKit identifies large aromatic macrocycles such as Porphyrins, while OpenBabel does not assign aromatic status to the bonds linking adjacent pyrrole rings. **(C)** Many differences in aromaticity assignment are due to RDKit's handling of exocyclic double bonds. **(D)** RDKit's handling of lone pairs and exocyclic double bonds causes it to assign aromaticity to the ester macrocycle in this molecule, however, OpenBabel identifies several distinct aromatic systems.

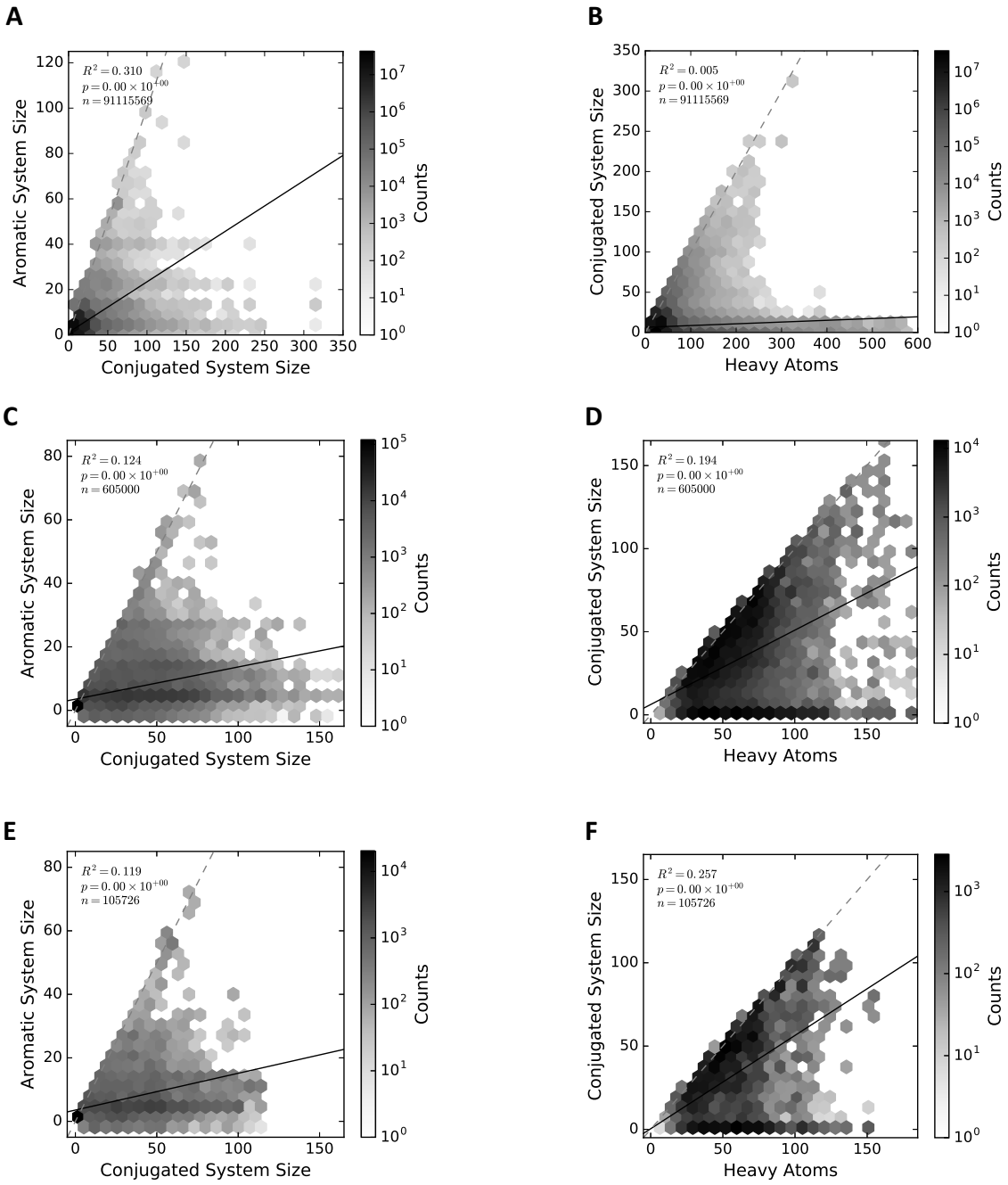

Figure S3: Uniform sample of molecules by aromatic system sizes and conjugated system sizes from PubChem. **(A)** A random sample of PubChem shows a strong bias towards small conjugated and aromatic system sizes and **(B)** small molecule sizes. **(C-F)** To represent the complete range of molecule sizes, aromatic system sizes, and conjugated system sizes, we selected molecules uniformly from our PubChem sample, selecting at least one molecule from each bin of histogram. In addition, we generated several adversarial versions of each molecule with one double bond changed to a single bond in the kekulized structure. The resulting distributions of aromatic and conjugated system sizes were similar between atoms in **(C, D)** the training set and **(E, F)** the test set.

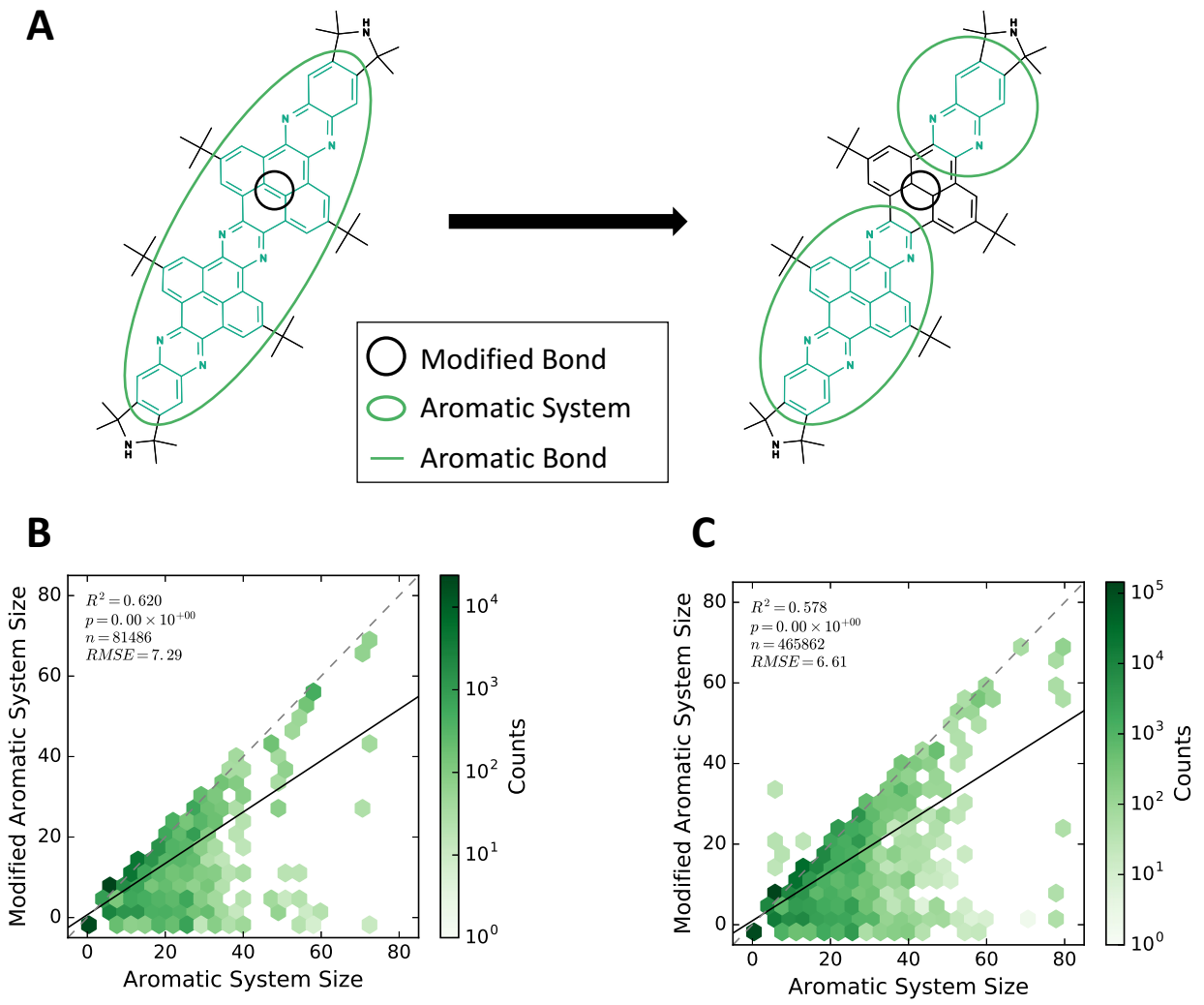

Figure S4: Adversarial examples split large aromatic systems into multiple smaller systems by modifying a double bond. (A) For each molecule in the dataset, multiple modified versions were generated, each with a double bond converted to single bond, and appropriate hydrogens added. For aromatic systems, two double bond modifications were chosen that led to (1) the greatest change in aromatic system size and (2) the average change in aromatic system size compared to other double bond modifications. **(B and C)** Modifying bonds in this way produced comparable decreases in the size of aromatic systems in both the training and test sets.

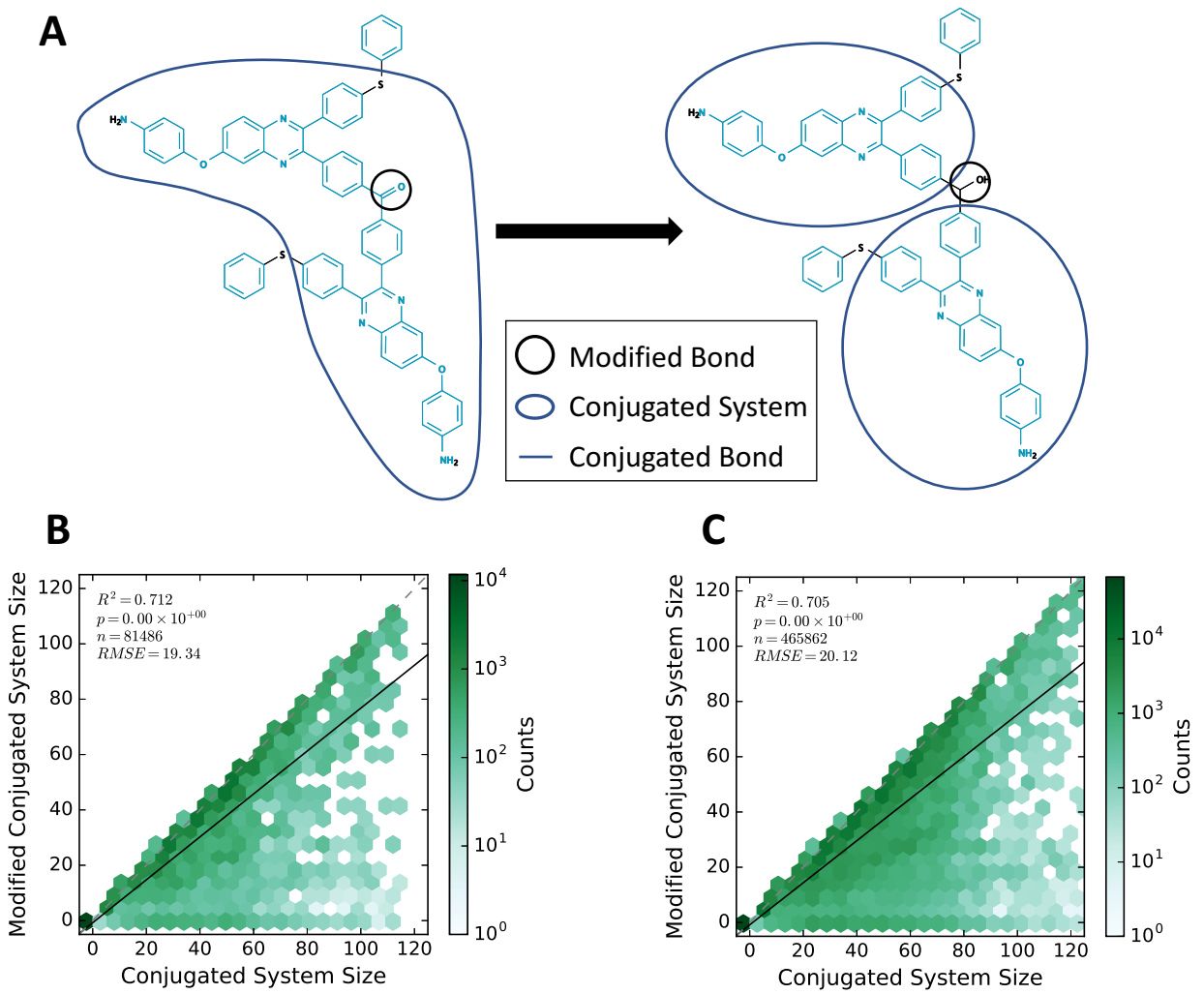

Figure S5: Adversarial examples split large conjugated systems into multiple smaller systems by modifying a double bond. (A) For each molecule in the dataset, multiple modified versions were generated, each with a double bond converted to single bond, and appropriate hydrogens added. For conjugated systems, two double bond modifications were chosen that led to (1) the greatest change in conjugated system size and (2) the average change in conjugated system size compared to other double bond modifications. **(B and C)** Modifying bonds in this way produced comparable decreases in the size of conjugated systems in both the training and test sets.

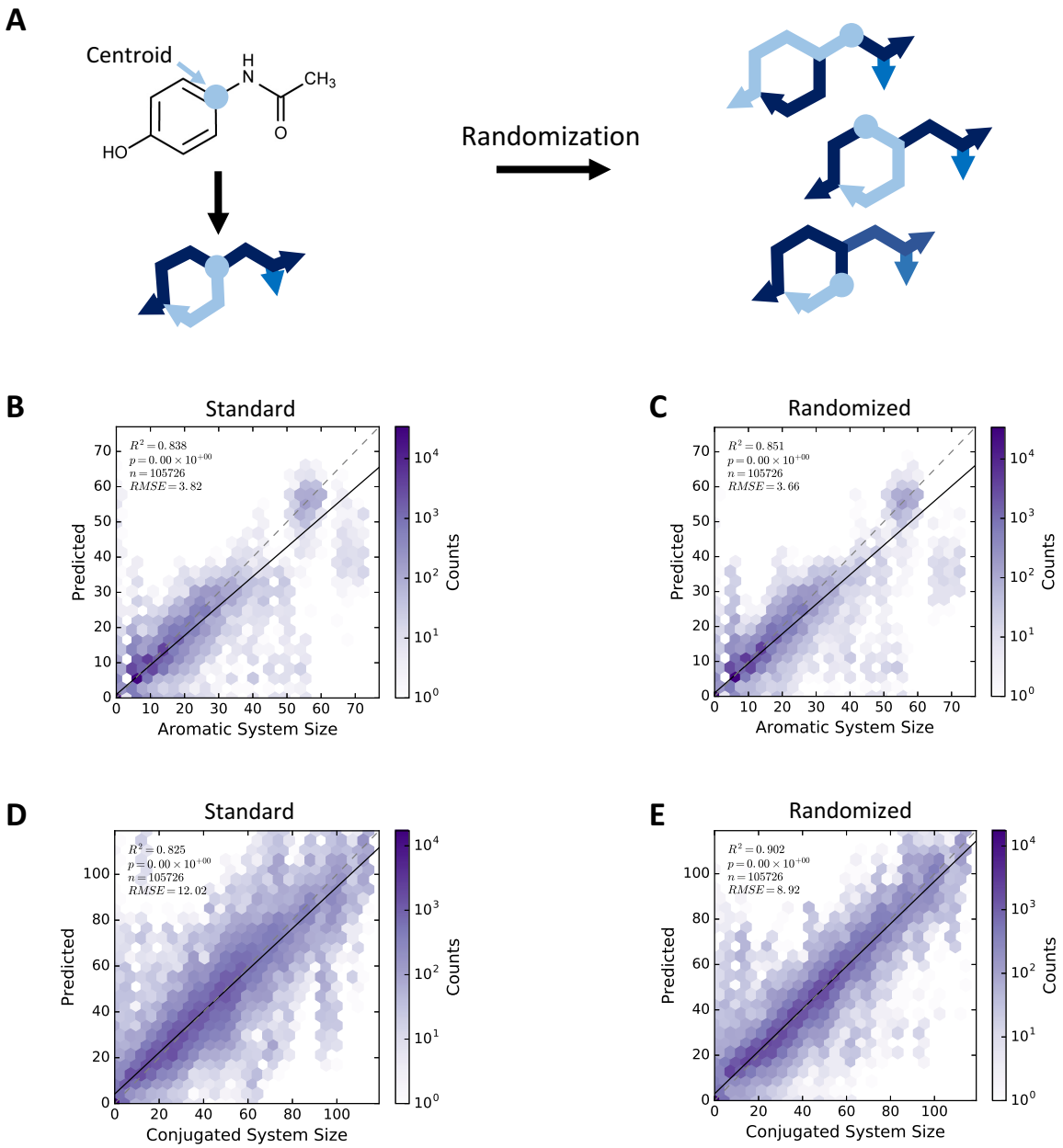

Figure S6: Randomization of BFS start point at training time improves validation accuracy when data is scarce. We evaluated the out-of-bag test accuracy of the N-WAVE model when trained with a significantly smaller training set of 1,948 molecules. **(A)** For our experiments, we chose the centroid as the starting point for our BFS to minimize the number of recursive computations required for each forward-backward pass. **(B and C)** However, randomly choosing a start point near the centroid during training slightly improves the generalization accuracy of the model for aromatic system size and **(D and E)** substantially improves the generalization accuracy for conjugated system size. This randomization serves as a regularization method analogous to random rotation, cropping and other image adjustments used to augment datasets in image classification problems.

Table S1: Baseline validation accuracy on all training targets. Bold indicates the best performance on each target. The performance of the WAVE variant architectures was superior to weave in 10 of 13 categories. Performance on Ring, AA, R3-8, BA and FBA was quantified by area under the receiver operator curve (higher is better). Performance on LRing, ArSS and CoSS was quantified by RMSE (lower is better). P = Parameter Count, Ring = Atom in ring, AA = Atom Aromatic, Rx = Atom in ring of size x, BA = Aromatic bond, FBA = Far bonds aromatic (not used in training, see text), LRing = Largest ring passing through atom, ArSS = Size of aromatic system of which atom is a member, CoSS = Size of conjugated system of which atom is a member.

Model	P	Ring	AA	R3	R4	R5	R6	R7	R8	BA	FBA	LRing	ArSS	CoSS
Neighbor	1,104	98.2	96.9	54.6	65.5	93.1	95.6	67.1	46.7	97.9	61.1	1.96	5.97	21.2
Weave	71,164	99.7	**99.7**	81.3	88.1	97.2	98.8	86.0	**85.9**	99.5	85.8	1.44	3.49	12.1
WAVE	50,289	**99.9**	99.5	91.2	81.5	98.3	99.2	83.0	73.2	99.6	88.2	1.38	3.07	4.39
N-WAVE	51,264	**99.9**	**99.7**	**95.3**	**92.1**	**99.7**	**99.5**	**87.4**	74.8	**99.8**	**91.6**	**1.34**	**2.67**	**3.94**

Table S2: Hyper-parameter values used for sweep experiments. For each model, each combination of parameter values was run three times. Among the three runs, the model with lowest training set error was used to evaluate test error for the parameter sweep.

	Parameter	Values
Weave Model	Number of Layers	2, 5, 10, 15, 20, 30, 50
	Atom Features	20, 30, 40
	Atom Hidden Layer Size	30, 40, 50
	Pair Features	3, 5, 10
	Pair Hidden Layer Size	5, 10, 15
Hybrid WAVE Model	Forward-Backward Passes	1, 2, 3, 4
	Neighborhood Distance	0, 3, 6
	Output Gate Hidden Layers	0, 1, 2
	Mix Gate Hidden Layers	0, 1, 2
	Atom Memory State Size	10, 20, 30

Table S3: Wave-like propagation performance is only weakly dependent on structure of recursive unit. The first row (B) shows the baseline model, and each defines a change to the baseline (black squares) that was assessed. The mix gate, which integrates data from multiple inputs, appears to be the only component with a strong effect on model accuracy. The weave baseline model accuracy is shown for comparison (bottom). Red colors represent worse validation error, while blue colors represent better validation error. PASS = the number of forward backward passes, SW = switch to new cell on backward pass, MG = mix gate, MI = Mix gate uses input from atom, MSM = mix softmax, MWS = mix weighted sum, MEM = mix edge mode (same = treat tree and cross edges the same, port = cross and tree edges have separate ports, fetch = port mode + fetch cross edge features from last layer), NA = network activation (elu = exponential linear unit, relu = rectified linear unit), N = neighborhood network of distance 3 at input layer, RG = read gate (not used by default), UG = update gate.

	Model Parameters											Validation RMSE		Training RMSE	
	PASS	SW	MG	MI	MSM	MWS	MEM	NA	N	RG	UG	Aromatic	Conjugated	Aromatic	Conjugated
B	3	yes	yes	yes	yes	yes	fetch	elu	yes	no	yes	2.67	3.94	1.43	2.56
1	**1**											0.78	7.61	1.14	7.56
2		**no**										0.44	0.91	0.55	1.22
3			**no**									0.41	1.77	0.50	1.45
4			**no**								**no**	0.30	2.09	0.69	2.19
5				**no**								0.05	0.65	0.15	0.29
6					**no**							0.18	0.65	0.36	0.27
7						**no**						0.04	1.42	0.14	0.70
8							**same**					0.26	1.02	0.46	1.15
9							**port**					0.04	0.21	0.06	-0.06
10								**relu**				0.17	0.49	0.26	0.41
11									**no**			0.10	1.06	0.22	0.60
12										**yes**		0.09	0.52	0.25	0.36
13											**no**	-0.06	0.21	0.03	0.08
	Baseline Weave Model (increase over N-WAVE baseline)											0.82	8.16	1.04	6.97

Other Supplementary Materials

Test and Training Set Molecules

We have included an archive zip file containing (1) the training set molecules and (2) the test set molecules. Molecule files are presented in SDF format,[3] which is readable by most commonly used chemical informatics toolkits (eg. OpenBabel, RDKit, Chemistry Development Kit, and others). Each molecule is labeled with the compound ID of the PubChem record from which it was derived (eg. 68679049). Adversarial examples are additionally labeled with the indices of the two atoms flanking the modified bond (eg. 68679049_modified_at_22_23).

References

(1) Lusci, A.; Pollastri, G.; Baldi, P. Deep Architectures and Deep Learning in Chemoinformatics: The Prediction of Aqueous Solubility for Drug-Like Molecules. J. Chem. Inf. Model. **2013**, 53, 1563–1575.

(2) Carleo, G.; Troyer, M. Solving the quantum many-body problem with artificial neural networks. Science **2017**, 355, 602–606.

(3) O'Boyle, N. M.; Banck, M.; James, C. A.; Morley, C.; Vandermeersch, T.; Hutchison, G. R. Open Babel: An open chemical toolbox. J. Cheminf. **2011**, 3, 33.