Predicting electronic properties of materials

Ilan Rosen (irosen) and Jason Qu (jayqu), with Jacob Marks (from CS229a)

(Dated: December 13, 2018)

We investigate the power of a variety of learning algorithms and physically motivated feature encodings to predict bandgaps on the JARVIS-DFT 3d dataset using only unit cell composition and relative atomic positions. We find that, defying intuition, a one-hot vector encoding of elemental composition outperforms Coulomb matrix-based encodings for metal-nonmetal classification and nonmetal bandgap size regression. Our final pipeline consisted of a random forest classifier, which obtained an F1-score of 0.767, and a neural network regressor, which achieved a root mean squared error of 0.924 eV.

I. INTRODUCTION

The application of machine learning (ML) techniques for understanding materials is a burgeoning field of research that has already seen success in academic and commercial settings. For example, ML methods have been used to improve estimates of electron wavefunctions, from which a material's optical and elastic properties can be computed. Such research, called high-throughput computational materials design, aims to identify materials likely to exhibit properties of interest (e.g., metallic materials with exceptional tensile strength or superconducting ceramics).[1] Once the enormous space of possible materials is restricted to a few such candidates, each candidate is synthesized and tested in a laboratory.

A major challenge in applying ML to materials research is the vast space of possible materials. Further, materials with similar compositions may exhibit wildly different observable characteristics. Training data exists for only some of these materials. In addition, it is unclear *a priori* which features are important to predict observable properties. Thus, feature engineering will be crucial for the success of our project.

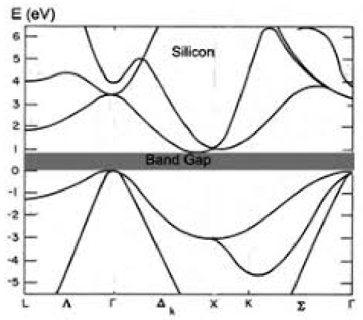


FIG. 1. Energy landscape of silicon—the bandgap is shaded.[2]

We will use ML to predict the electronic bandgap of inorganic solids. Roughly, the bandgap, shown in Fig. 1, is an electronic property of a material describing the energy required to add an electron to the material. Materials with no gap are conductors, whereas materials with large gaps are insulators. Materials with small gaps are called semiconductors and can be made to either insulate or conduct electricity by small changes to their chemistry or electrical environment.

Bandgaps are difficult to predict based on material compositions. For example, VO_x can be either conducting or insulating depending on the value of x and the temperature. Yet obtaining accurate predictions of gaps is crucial to the development of improved electronic materials. Currently, the best technique to predict bandgaps is density functional theory (DFT). DFT, developed in 1964 by Hohenberg, Kohn, and Sham, makes *ab initio* (from first principles) calculations of material properties, meaning it uses no phenomenological parameters.[3,4] It is of the best computational tools for the prediction of material properties, but is extremely computational expensive.[5]

Here, we employ ML to overcome the computational limitations of DFT. Our objective is to develop a learning algorithm to provide fast and accurate predictions of electronic bandgaps, bypassing the need for expensive or inaccurate DFT computations. Our model will make predictions based on the elemental compositions and crystal structures of materials.

II. RELATED WORKS

Multiple efforts have been made to predict electronic properties with ML. These works motivated our use of the Coulomb Matrix as a feature encoding,[6,7] and our choice of root mean squared error (RMSE) as the error metric against which to optimize hyper-parameters on our validation set.[8] While previous works have predicted bandgaps using ML, these studies either considered a particular type of material (such as organic compounds or particular crystalline structures) or used various additional electronic properties as input features, reducing their real-world utility.[9–11] Our project approaches the broader task of predicting gap size *across* material classes, and *without* additional electronic property data. As such, we anticipate higher error rates than in other works, but a more powerful and broadly applicable model overall.

III. DATASET

Our model is developed using JARVIS (Joint Automated Repository for Various Integrated Systems), a database of materials properties that are computationally generated using DFT.[12] Of the 25,923 three-dimensional inorganic solids in the database, 23,455 list the calculated bandgap. While metals truly have zero (or negligible) gap, every material in the JARVIS dataset was labeled with a finite, albeit often quite small, gap—an error associated with DFT calculations. In literature, the smallest known semiconductor bandgap is 0.015 eV, so we set our cutoff at 0.01 eV. Using this cutoff, the 3d dataset contained 14752 nonmetals and 8703 metals.

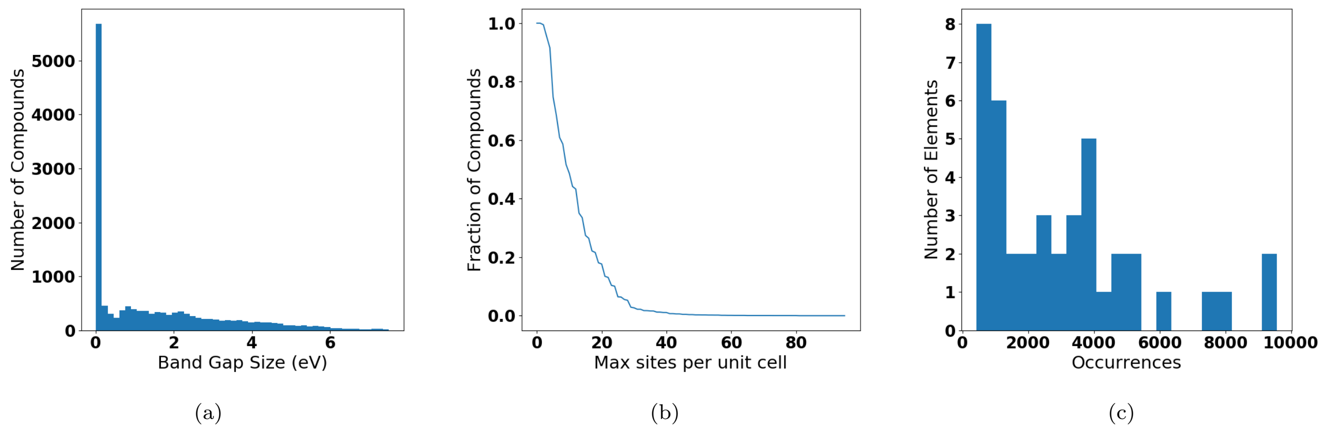


FIG. 2. Characteristics of our full DFT generated dataset. (a) Histogram of bandgap sizes (eV) for nonmetals. (b) Cumulative distribution function of compounds vs. maximum number of atoms per unit cell. (c) Histogram of element occurrences in all compounds.

In the dataset, certain elements occur more frequently than others, and different compounds contained different numbers of atoms per unit cell. During our experiments, we examined the effect of limiting our dataset to compounds with elements that occurred in at least m training examples, as well as limiting to compounds with N or fewer sites per unit cell.

The full dataset was randomly partitioned into 60% and 20% training and development splits, for developing the models, and a 20% testing split, that is withheld and used to benchmark the algorithm's performance. The large sizes of the development and test splits are necessary to adequately sample across the huge variety of material types.

It is crucial to develop a representation of the crystal structure information that simultaneously preserves the vital physics of the problem and serves as an efficient set of features for the machine learning algorithm. We considered a number of feature representations:

One-hot vector: The simplest encoding we propose is a one-hot vector representing a material's elemental composition. The dataset cumulatively uses 94 elements, so, in this representation, the n^{th} element of a length-94 vector contains the fraction of the unit cell atoms comprised of element number n. For instance, the (fictional) compound H_1Al_2 is represented as the vector $(1/3, 0, 2/3, 0, 0, ...)$ since H and Al are the element numbers one and three, respectively. This representation of features has the advantage that a neural network can learn about the properties of each element independently.

However, the one-hot representation has numerous disadvantages. It lacks any structural data about the crystal, which is physically vital—e.g., diamond is an insulator while graphite is a conductor, although both are crystals of pure carbon. Furthermore, the one-hot vector is a 94 dimensional vector; such a large feature vector can result in a high-variance model. Finally, heavy elements, such as Eu, are rare and only appear a few times in the training set. Therefore, the model will be under-trained with respect to a large proportion of the features, resulting in a high-variance model.

Group one-hot vector: Rather than introducing a new feature for each of the 94 elements, we propose a group one-hot encoding, which uses an 18 dimensional one-hot vector having one component for each group in the periodic table. Much of the relevant physics to material properties is contained in the number of valence electrons of elements, a common quantity among elements of like group. At the potential expense of increased bias, this representation should reduce the model's variance relative to the one-hot encoding, as it associates rare elements with common elements of the same group.

Coulomb matrix: To create a more complex feature set, we would like to add structural information about the crystals. The real-space coordinates of the atomic positions, however, are degenerate in choice of coordinate and therefore are themselves a poor set of features. The Coulomb matrix is a coordinate invariant representation of this information. The Coulomb matrix expresses the potential energy between pairs of atoms in the crystal, which depends on their pairwise distance and atomic numbers. The Coulomb matrix C is a symmetric matrix defined element-wise as

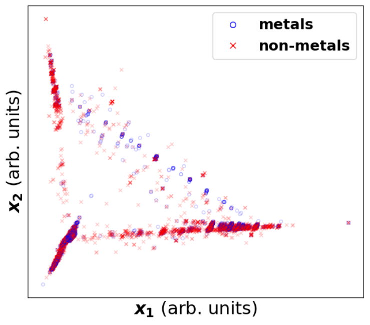
$$C_{ii} = \frac{1}{2} Z_i^{2.4}, \quad C_{ij} = \frac{Z_i Z_j}{|r_i - r_j|}$$

for $i \neq j$, where Z_i is the nuclear charge on the ith lattice site and r_i is its position[6]. The Coulomb matrix, however, lacks a specific information of the crystal's constituent elements and only describes the atomic charges. Therefore, the neural net cannot learn properties associated with individual elements (for instance, Ar and K have very different electronic properties, despite their charges Z differing only by 1).

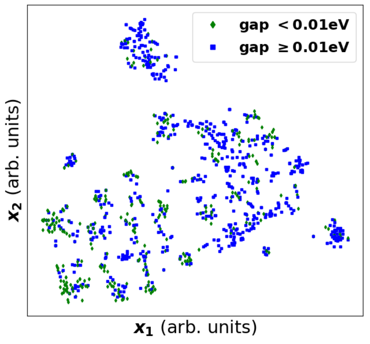
A second drawback of the Coulomb matrix is that its size is the number of lattice sites in the compound, which varies between compounds. We assuage the latter issue by choosing all Coulomb matrices to be the a certain size N; smaller Coulomb matrices are padded with zeros and crystals with too many atomic sites are disregarded.

As the Coulomb matrix is symmetric, we include only its upper triangular entries as features, reducing the number of features from N to $(N+1)/2$. The upper triangular components are reshaped into a vector-valued feature.

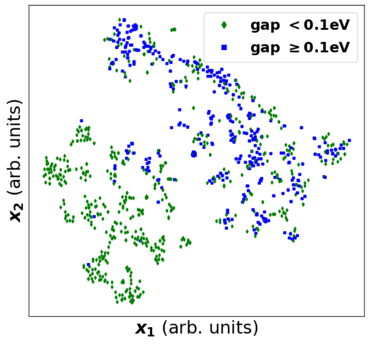
Coulomb matrix Extensions: While the Coulomb matrix is invariant in the choice of real-space coordinates, it is not invariant under permutations of its rows and columns, although such permutations do not alter the material represented by the matrix. We employed two methods to alleviate



(a)



(b)



(c)

FIG. 3. Visualizations of training data (a) First two principle components of element one-hot encoding. (b) tSNE of group one-hot encoding with gap size cutoff 0.01 eV. (c) tSNE of group one-hot encoding with gap size cutoff 0.1 eV.

this issue. First, we used the singular values of the Coulomb matrix as features, listed in order of decreasing value. The singular values are invariant to permutations of row-column pairs. While they do not contain all of the information from the Coulomb matrix, we expect training on the singular values to give more consistent results. Second, used Coulomb matrices as features directly, but augmented the training set with the Coulomb matrices of training examples under random permutations of row-column pairs. This method preserves all the information contained in the original Coulomb matrix, at the expense of a dramatically increased memory and computation power requirement.

Finally, to present our models with the maximum amount of information, we used an encoding including both the Coulomb matrix and the one-hot representations atom's groups, as well as an encoding including both the singular values of the Coulomb matrix and the one-hot representation of each element.

To visualize the dataset, we performed principle component analysis (PCA) on the dataset under the element one-hot feature encoding (Fig. 3). This analysis reveals an underlying structure to the dataset; this structure, however, is uncorrelated with the metallicity of the examples. We also visualized the group one-hot encoding using t-distributed stochastic neighbor embedding (tSNE). The resulting plots partially differentiated materials with gaps less than versus greater than 0.1 eV, but did not distinguish materials when the cutoff was set to 0.01 eV—the threshold we selected for metal/nonmetal classification, based on literature. These observations affirm our expectation of the difficulty of classifying metallicity.

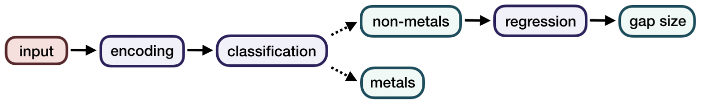


FIG. 4. Pipeline for model predictions.

IV. METHODS

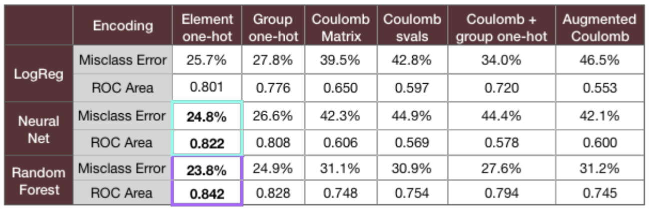
We developed a pipeline to transform the inputs to our model, the crystal structures of materials, into useful predictions. The crystal structures are first encoded into appropriate feature representations. Next, a classification stage uses these features to predict which materials are metals and which are nonmetals. The predicted nonmetals are then sent to a regression stage, which predicts their bandgap.

The classification and regression learning models were trained on the training set and optimized using the development set. Optimization included testing different learning algorithms on each feature encoding, and then tuning hyperparameters on the best-performing encoding-algorithm combinations. To prohibit the regression stage from being biased on the performance of the classification stage, the regression stage was trained and developed on the set of true nonmetals, not the set of predicted nonmetals (the latter inevitably includes misclassified metals and omits misclassified nonmetals). The performance of the full pipeline was characterized using the withheld testing set.

We developed code to parse and encode the dataset in Python. We used a combination of self-develop machine learning algorithm implementations, written in Python, and methods from Python's scikit-learn package. We classified materials as metals or nonmetals classification using logistic regression, neural networks (NNs), and random forest (RF) classifiers. We predicted the size of the gap using linear regression, NNs, and RF regressors. We describe these models, taking m as the number of training examples, n as the number of input features, $X \in \mathcal{R}^{m \times n+1}$ as a matrix of input features, and $y \in \mathcal{R}^m$ as the output vector of bandgap sizes. θ denotes parameters that the model learns, and takes different sizes in the different algorithms. J represents the cost function, and λ is a regularization parameter that prevents the models from over-fitting the training set by enforcing that the magnitudes of the learned parameters remain small.

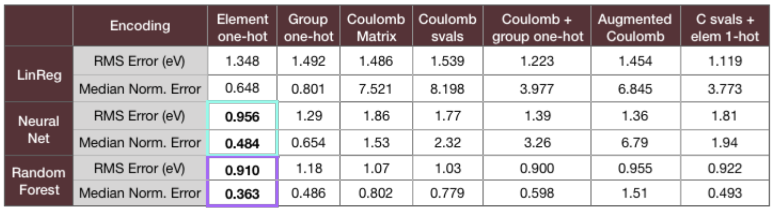
Linear Regression: The hypothesis is linear in the input features: $h_\theta(X) = X\theta$. The cost function is $J(\theta) = \frac{1}{2m}(X\theta - y)^T(X\theta - y) + \frac{\lambda}{2m}\theta_1^2$, where $\theta \in \mathcal{R}^{n+1}$ is a vector of parameters that the model learns, with the first component being a bias term, and θ_1 denoting a vector of all the components of θ excluding the bias term.

Logistic Regression: The hypothesis is now given by a sigmoid, which introduces nonlinearity: $h_\theta(X) = \frac{1}{1+e^{-X\theta}}$ applied element-wise. $h_\theta \in [0,1]$, with examples classified as positive ($\hat{y} = 1$) if $h_\theta \geq 0.5$ and negative ($\hat{y} = 0$) otherwise. The cost function is $J(\theta) = -\frac{1}{m}\big(y \log(h_\theta(X)) + (1 -$



Encoding		Element one-hot	Group one-hot	Coulomb Matrix	Coulomb svals	Coulomb + group one-hot	Augmented Coulomb
LogReg	Misclass Error	25.7%	27.8%	39.5%	42.8%	34.0%	46.5%
	ROC Area	0.801	0.776	0.650	0.597	0.720	0.553
Neural Net	Misclass Error	**24.8%**	26.6%	42.3%	44.9%	44.4%	42.1%
	ROC Area	**0.822**	0.808	0.606	0.569	0.578	0.600
Random Forest	Misclass Error	**23.8%**	24.9%	31.1%	30.9%	27.6%	31.2%
	ROC Area	**0.842**	0.828	0.748	0.754	0.794	0.745

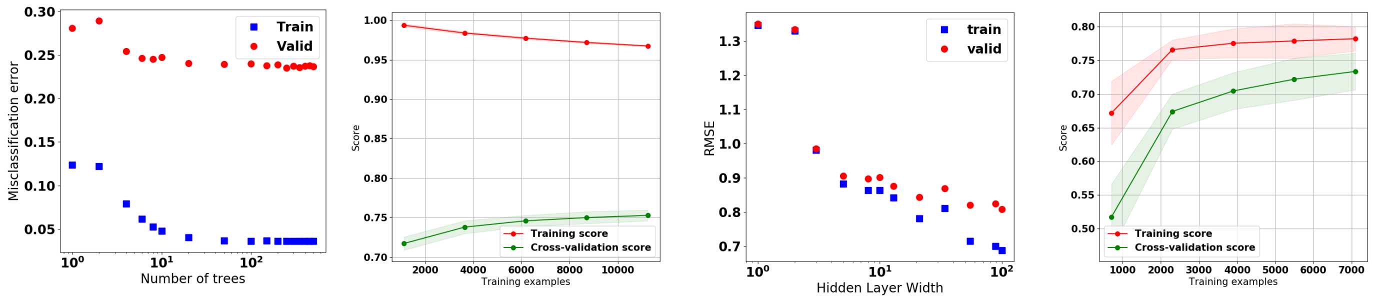
(a)



Encoding		Element one-hot	Group one-hot	Coulomb Matrix	Coulomb svals	Coulomb + group one-hot	Augmented Coulomb	C svals + elem 1-hot
LinReg	RMS Error (eV)	1.348	1.492	1.486	1.539	1.223	1.454	1.119
	Median Norm. Error	0.648	0.801	7.521	8.198	3.977	6.845	3.773
Neural Net	RMS Error (eV)	**0.956**	1.29	1.86	1.77	1.39	1.36	1.81
	Median Norm. Error	**0.484**	0.654	1.53	2.32	3.26	6.79	1.94
Random Forest	RMS Error (eV)	**0.910**	1.18	1.07	1.03	0.900	0.955	0.922
	Median Norm. Error	**0.363**	0.486	0.802	0.779	0.598	1.51	0.493

(b)

FIG. 5. Validation performances of the different learning algorithms using different input feature encodings. (a) Metal/nonmetal classification. The error metrics are misclassification error and the error under the receiver operating curve. (b) Regression to predict the bandgap value. The error metrics are RMSE and median normalized error. NNs one layer of width 10; RFs have 200 trees. The best performing models, which are later more carefully tuned, are highlighted.



(a) (b) (c) (d)

FIG. 6. (a) Hyperparameter tuning for the RF classifier. The misclassification error is shown versus the number of trees. (b) Learning curve of the RF classifier. The F1 score is shown versus the number of training examples. (c) Hyperparameter tuning for the NN regressor. The RMSE is shown versus the layer width. (d) Learning curves of the NN regressor. The R^2 score is shown the number of training examples, respectively. (b) and (d) The training scores (red) and cross validation (CV) scores (green) curves are shown. The 5-fold CV score is the mean of 100 splitting iterations. Shading indicates one standard deviation of the score mean.

$y)\log(1-h_\theta(X))\big)+\frac{\lambda}{2m}\theta_1^2$.

NN: Neural Networks are nonlinear models that can approximate high variance data. They can be used for either classification or regression, depending on the loss function. The number of hidden layers, number of neurons per layer, and the activation function are all model choices. For binary classification, the cost is given by $J(\theta)=\sum_{i=1}^m -\frac{1}{m}\left(y^{(i)}\log\hat{y}^{(i)}+(1-y^{(i)})\log\left(1-\hat{y}^{(i)}\right)\right)+\frac{\lambda}{2m}\theta_1^2$, where $\hat{y}=a_2$ is the output of the NN. For real-valued regression, the cost function is given by $J(\theta)=\frac{1}{2m}\left(\hat{y}^{(i)}-y^{(i)}\right)^2+\frac{\lambda}{2m}\theta_1^2$. We used neurons with rectified linear unit (ReLU) activation $a=g(z)=\max(0,z)$. The output neuron of the NN classifier is a sigmoid, and that of the NN regressor is $a=g(z)=z$.

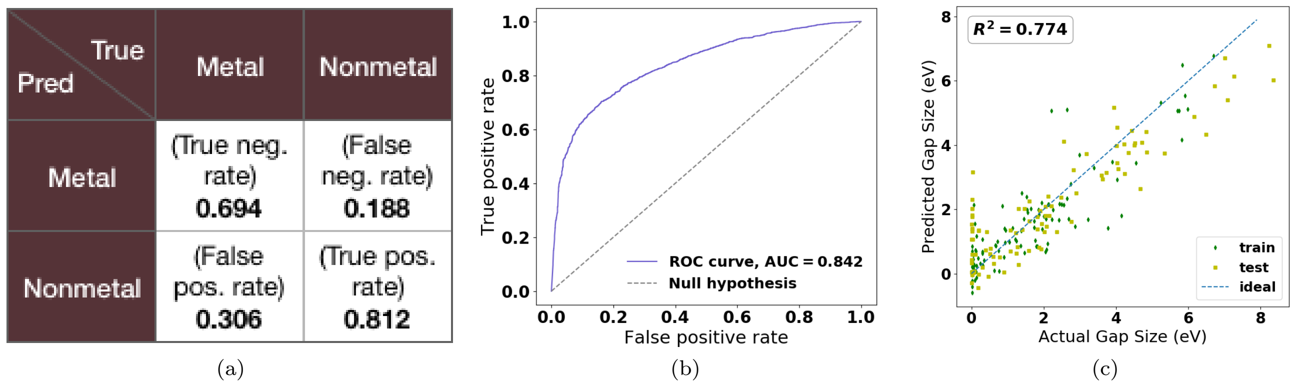
Random forests: RFs are averages across independently grown trees and are less prone to over-fitting than individual trees. Decision trees are classifiers that repeatedly partition the data into branches chosen to minimize the Gini impurity $I_P = 1-\sum_{c=0,1}p_c^2$ where p_c is the probability that an example in the branch is of class c. RF classifiers take the mode over many individual trees. Regression trees choose branches to minimize the total variance within branches $I_V=\frac{1}{2|S|}\sum_{i,j\in S}(x_i-x_j)^2$. RF regressors take the mean over many individual trees. The parameter “max features” determines the number of random features considered when determining each split.

Metrics: Along with the misclassification error, we quantified the performance of the classifier by the area under the receiver operator characteristic (ROC). The ROC displays true positive rate versus the false positive rate as the classification threshold is varied. We also considered the F1-score $F_1=\frac{2TP}{2TP+FP+TN}$ where T(F)P(N) represents the number of true (false) positives (negatives). We quantified the performance of the regressor using the median normalized error and the RMSE $\sqrt{\frac{1}{m}\sum_i^m(y_i-\hat{y}_i)}$, where $\hat{y}$ is the predicted value of the bandgap. We also considered the R^2 score $1-\frac{\sum_i^m(y_i-\hat{y}_i)^2}{\sum_i^m(y_i-\bar{y})^2}$, where $\bar{y}=\frac{1}{m}\sum_i^m y_i$.

V. RESULTS

All learning models for the classification and regression problems were tested on all feature encodings (Fig. 5). The element one-hot encoding consistently outperformed all other encodings. A RF classifier was chosen for the first stage, and a NN regressor, for the second. Optimal hyperparameters were chosen for both (Fig. 6). The RF classifer overfit the training set as the number of trees increased, though its validation accuracy did not decrease. We ultimately chose a RF classifier with 200 trees and 8 maximum features per split, and a single layer NN regressor with width 10, ReLU activation, learning rate $1e-3$, and regularization parameter $\lambda = 0.01$. The element one-hot encoding was used as the feature vector for both stages.

Having finalized the full prediction pipeline, we characterized its performance on the test set (Fig. 7). The RF



True / Pred	Metal	Nonmetal
Metal	(True neg. rate) **0.694**	(False neg. rate) **0.188**
Nonmetal	(False pos. rate) **0.306**	(True pos. rate) **0.812**

FIG. 7. (a) Truth table of the RF classifier's predictions on the test set. (b) The receiver operating characteristic of the RF classifier. (c) True vs. predicted gap size from the NN regressor for 100 randomly selected examples from the training and test sets.

classifier achieved a misclassification accuracy of 23.2% and an F1 score of 0.767. The regression stage (tested on the set of true AND predicted nonmetals) achieved an RMSE of 0.924 eV, a median error of 0.364, and an R^2 score of 0.774; its performance is visualized in Fig. 7(c).

VI. DISCUSSION

The learning curve in Fig. 6(b) illustrates that the classifier performs substantially better on the training set than the validation set, indicating high classification variance. This suggests that the model could be improved by reducing the number of input features, or by increasing the number of training examples. The group one-hot encoding proved to not be an effective reduction of the feature set. A feature set generated by applying a dimensionality reduction algorithm such as PCA to the element one-hot vector may perform better.

Fig. 6(d) suggests that the error in the regression stage is dominated by bias. Further effort towards feature engineering will mitigate this issue.

Both the classifier and the regressor would benefit from a larger training set. The augmented Coulomb matrix encoding increased the number of training examples by a factor of 10, but resulted in no performance benefit at a high computational expense. Incorporating additional DFT databases into the training dataset may improve our predictions.

In literature, 0.01 eV is generally used as the cutoff between metals and nonmetals. However, nearly half of the nonmetals in the dataset had gaps between 0.01 eV and 0.1 eV (Fig. 1). Many of these materials are actually metals that DFT has inaccurately characterized. Changing the cutoff to 0.1 eV increases the classification accuracy of our model to 89.4%. In addition, the regressor achieves a substantially smaller median normalized error of 0.235 (the RMSE is essentially unaffected).

VII. CONCLUSION

Our model's accuracy is sufficient to provide useful predictions for high-throughput materials screening. The leading performance of the element one-hot encoding is surprising as it includes no information about materials' crystal structures. The group one-hot encoding performed well at classification but poorly at regression. We explain this discrepancy by noting that the valence of an atom (which is given by it's group) is important for predicting metallicity, but that the value of the bandgap depends on electronic potentials, information that the group one-hot encoding lacks. Adding the singular values of the Coulomb matrix as features supplies this information, and correspondingly increases the regressor's accuracy.

Much of our model's error came from errors in the training dataset for incorrectly labeled examples with gaps between 0.01 eV and 0.1 eV. To correct for this issue, we propose to augment the DFT-generated dataset with experimentally obtained values for small bandgap semiconductors; these examples would be heavily weighted in the training algorithm.

We propose future work to add features representing element's electronic attributes, such as the orbital character of their valence electrons, to increase the complexity of the features. Further, to better represent structural information, we propose to use a unsupervised learning (such as the mixture of Gaussians model) to transform the atomic positions into categories of crystal structures based on their symmetries. Including these features will allow a neural network to discriminate between categories of materials.

VIII. CONTRIBUTIONS

All authors contributed equally to all components of this work.

[1] S. Curtarolo, G. L. W. Hart, M. B. Nardelli, N. Mingo, S. Sanvito, and O. Levy, Nature Materials **12**, 191 EP (2013).
[2] I. Delhi, "Fundamental concepts of semiconductors," (2013).
[3] P. Hohenberg and W. Kohn, Phys. Rev. **136**, B864 (1964).
[4] W. Kohn and L. J. Sham, Phys. Rev. **140**, A1133 (1965).
[5] A. Parrill and K. Lipkowitz, *Reviews in Computational Chemistry*, Vol. 31 (Wiley, 2018).

[6] K. T. Schütt, H. Glawe, F. Brockherde, A. Sanna, K. R. Müller, and E. K. U. Gross, Phys. Rev. B **89**, 205118 (2014).

[7] M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. von Lilienfeld, Phys. Rev. Lett. **108**, 058301 (2012).

[8] A. C. Rajan, A. Mishra, S. Satsangi, R. Vaish, H. Mizuseki, K.-R. Lee, and A. K. Singh, Chemistry of Materials **30**, 4031 (2018), https://doi.org/10.1021/acs.chemmater.8b00686.

[9] G. Pilania, A. Mannodi-Kanakkithodi, B. P. Uberuaga, R. Ramprasad, J. E. Gubernatis, and T. Lookman, Scientific Reports **6**, 19375 EP (2016).

[10] Y. Zhuo, A. Mansouri Tehrani, and J. Brgoch, The Journal of Physical Chemistry Letters **9**, 1668 (2018).

[11] A. C. Rajan, A. Mishra, S. Satsangi, R. Vaish, H. Mizuseki, K.-R. Lee, and A. K. Singh, *Chemistry of Materials*, Chemistry of Materials **30**, 4031 (2018).

[12] K. Choudhary, I. Kalish, R. Beams, and F. Tavazza, Scientific Reports **7**, 5179 (2017).