THE JOURNAL OF
PHYSICAL CHEMISTRY A

pubs.acs.org/JPCA

Article

Lattice Thermal Conductivity Prediction Using Symbolic Regression and Machine Learning

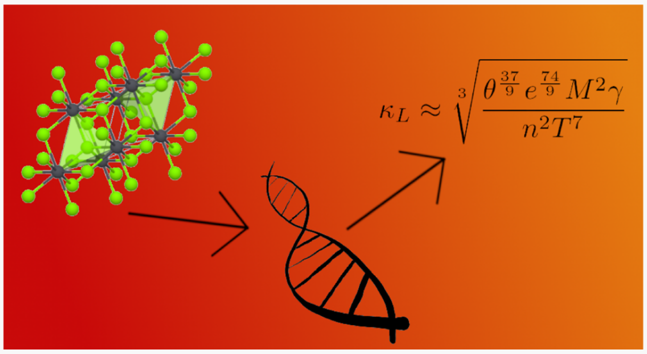
Christian Loftis, Kunpeng Yuan, Yong Zhao, Ming Hu,* and Jianjun Hu*

Cite This: *J. Phys. Chem. A* 2021, 125, 435−450

Read Online

ACCESS | Metrics & More | Article Recommendations | SI Supporting Information

ABSTRACT: Prediction models of lattice thermal conductivity (κ_L) have wide applications in the discovery of thermoelectrics, thermal barrier coatings, and thermal management of semiconductors. However, κ_L is notoriously difficult to predict. Although classic models such as the Debye−Callaway model and the Slack model have been used to approximate the κ_L of inorganic compounds, their accuracy is far from being satisfactory. Herein we propose a genetic programming-based symbolic regression (SR) approach for finding analytical κ_L models and compare them with multilayer perceptron neural networks and random forest regression models using a hybrid cross-validation (CV) approach including both *K*-fold CV and holdout validation. Four formulae have been discovered by our SR approach that outperform the Slack formula as evaluated on our dataset. Through the analysis of our models' performance and the formulae generated, we found that the trained formulae successfully reproduce the correct physical law that governs the lattice thermal conductivity of materials. We also systematically show that currently extrapolative prediction over datasets with different distributions as the training set remains to be a big challenge for both SR and machine learning-based prediction models.


$\kappa_L \approx \sqrt[3]{\frac{\theta^{\frac{37}{9}} e^{\frac{74}{9}} M^2 \gamma}{n^2 T^7}}$

1. INTRODUCTION

Having the capability to predict lattice thermal conductivity (κ_L) of a crystalline material based on its composition and structure information has wide applications in new materials discovery and thus has received noticeable attention in the thermodynamics field.[1−3] Prediction enables materials scientists to screen materials with desired κ_L without having to synthesize the materials first for testing. The advantages of materials with both high and low κ_L abound. For example, materials with high κ_L are desirable for conducting heat, and their uses range from being used for coolant pipes in nuclear power plants to being used for heat sinks. κ_L is especially important for semiconductors, whose electrical resistance increase as their temperature falls. Optimizing for thermal conductivity independently of electron conductivity enables materials researchers to create electrical insulators that conduct heat well or, inversely, to create electrical conductors that do not transfer heat well. Possessing this degree of control over a material's conductivity (both thermal and electric) will allow researchers to synthesize materials for use in electronics that can transmit electricity easily yet conduct less heat than other materials with the same electrical conductivity. This leads to electronics that do not overheat as quickly, despite high transistor density. Slack and Morelli state that "its manipulation and control have impacted an enormous variety of technical applications, including thermal management of mechanical, electrical, chemical, and nuclear systems; thermal barriers and thermal insulation materials; more efficient thermoelectric materials; and sensors and transducers."[1]

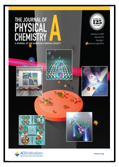
The thermodynamics research field has contributed several analytical models for calculating lattice thermal conductivity of materials including the well-known Slack model (Formula 1)[1] and Debye−Callaway Model (Formula 2).[2]

$$\kappa_L = A \cdot \frac{(\theta e)^3 M \sqrt[3]{V n_p}}{n^{4/3} T \gamma_a^{\ 2}} \tag{1}$$

$$\kappa_{L,tot} = A_1 \frac{M v_s^{\ 3}}{T V^{2/3} n^{1/3}} + A_2 \frac{v_s}{V^{2/3}} \left(1 - \frac{1}{n^{2/3}}\right) \tag{2}$$

Although these models are insightful, comparison with experimentally measured thermal conductivity has indicated that there is still plenty of room for improvement.[4−6] The models have also been shown to be less accurate than machine learning (ML) models that have been developed to predict κ_L.[2,3]

Received: September 4, 2020
Revised: November 19, 2020
Published: December 23, 2020



https://dx.doi.org/10.1021/acs.jpca.0c08103

Several recent studies have applied ML methods for thermal conductivity prediction.[3,7−14] Chen et al.[3] propose a Gaussian Process Regression combined with feature engineering by recursive feature elimination and random forest (RF)-based feature selection for LTC prediction. When applied to the small data set of 100 samples, they report a performance of R_2 0.93 when trained on 76 samples and tested on 19 samples. However, this result is questionable and may be due to the high redundancy/similarity of the samples. Although Chen et al. have certainly curated a diverse dataset and implemented cross-validation (CV) to discourage overfitting, it is impossible to completely eliminate—and it cannot be ignored that overfitting is a significant risk to small datasets. To improve the generalization performance, Juneja and Singh[8] proposed a localized regression-based patchwork kriging approach with elemental and structural descriptors for κ_L prediction. When applied to a dataset of 2838 materials, higher transferability has been achieved. Wan et al.[11] applied the XGBoost algorithm based on the descriptors of crystal structural and compositional information to κ_L prediction. Two geometric descriptors have also been shown to be closely related to thermal conductivities.[12] To address the issue of limited materials with annotated κ_L, a shotgun transfer learning approach has been proposed and applied to a small κ_L dataset of 95 samples. A major improvement of κ_L prediction comes from Zhu et al.'s work,[9] in which both graph convolution network and RF with elemental and structure features have been used to derive a prediction model over a much larger dataset with 2700 training samples. However, all these studies have not evaluated the real extrapolation performance.[15] Although these ML approaches have demonstrated themselves to be suited to predicting κ_L, they are unfortunately limited by nature in the insight that they can provide to the thermal science community as most of the ML models are essentially based on interpolation.

This study seeks to bridge the gap between the analytical models and ML models for κ_L prediction by exploring three types of models by focusing on the extrapolative prediction or generalization performance of three types of prediction models. Our first model is based on genetic programming (GP) symbolic regression (SR), which is an evolutionary algorithm that can generate formulae to map ordinal material properties to κ_L. The second model is a deep neural network model using a multilayer perceptron (MLP) powered by the Adam optimizer to predict κ_L by analyzing both the linear and nonlinear relationships in the data. Finally, the third model uses the RF regressor (RFR), a traditional ML method that has been shown to be effective in predicting κ_L.[3,9,16] We derive several formulae using the SR method that outperforms the Slack formula on our test dataset. In addition, analysis of our models' performance and formulae highlight interesting variable relationships to κ_L calculation and prediction, which showed the advantage of interpretable models of SR.

The SR models in this study take three forms. The first form, referred to as GP1, uses a limited function set with the intention of discovering models similar to the classic Slack model. The second, GP2, is provided with a richer function set to find formulae that are better than the Slack formula or are otherwise analytically distinct. Finally, the third model is a proof-of-concept model that illustrates the effectiveness of the SR methodology by attempting to rediscover the Slack formula from raw data points. SR is significant to the estimation of κ_L as it allows us to find and understand physical insights that may have otherwise been overlooked by physicists. Because the algorithm produces formulae purely from data points, there is no bias to the algorithm from any human or field specific knowledge. Through studying the formulae produced by the algorithm, we hope to uncover new physical insights into κ_L approximation.

2. METHODS

2.1. Dataset and Features. Each model is provided with the same set of descriptors (Table 1), with the exception that the SR models are unable to use the space group variable. This is due to the nature of the SR models, which require numeric fields that can be used as variables inside of formulae. In order to mitigate issues with fitting the models to the data, all materials with observed κ_L above 120 are recognized as outliers and thus trimmed from the dataset used for training and validation. The value 120 is chosen because the dataset contained a much higher concentration of data points with κ_L immediately below 120 than those above 120. The distribution and range of the dataset's κ_L values can be seen in Figure 1. In total, there are 347 samples. We have utilized two sampling techniques, which are explained in more detail in Section 3.1. The first method is randomly sampling data points directly from the dataset. The second method utilized is extrapolation sampling, which samples materials from different regions on the κ_L spectrum, thus showing the performance of the models when trained on different types of samples.

Table 1. List of Descriptors and Their Respective Definitions

variable symbol	definition
V	volume per atom
T	temperature (constant: 300 K)
M	average atomic mass
N	total number of atoms in unit cell
n_p	total number of atoms in primitive cell
B	bulk modulus calculated from C_{ij}
G	shear modulus calculated from C_{ij}
E	Young's modulus
ν	Poisson's ratio
H	estimated hardness
B'	$(\delta B/\delta V)$
G'	$(\delta G/\delta V)$
P	mass density
ν_L	sound velocities of the longitude
ν_S	sound velocities of the shear
ν_a	corresponding average velocity
Θ_D	Debye temperature
γ_L	longitude acoustic Grüneisen parameters
γ_S	shear acoustic Grüneisen parameters
γ_a	average acoustic Grüneisen parameters
A	empirical parameter

The dataset is collected from several published papers.[17−22] Most of the materials are half-Heusler compounds, oxide and fluoride perovskite, rocksalt-type, zincblende-type, and wurtzite-type compounds, and some thermoelectric materials. To prepare the dataset, all the first-principles calculations are carried out based on density functional theory (DFT) as implemented in the Vienna Ab initio Simulation Package.[23] The projector-augmented wave pseudopotentials[24] are used to describe the interaction among atoms, and the generalized gradient approximation in the Perdew−Burke−Ernzerhof[25]

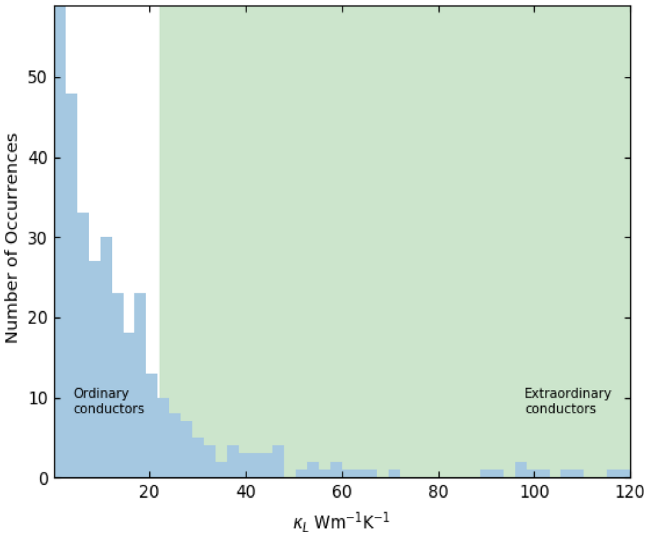


Figure 1. Histogram depicting the range and distribution of the κ_L of the materials in the dataset, with each bar representing a precision of 2.4 W m^{-1} K^{-1}. The X axis displays the κ_L of the materials, while the Y axis shows how many samples in the dataset fall within this range. The top 20% of the κ_L range (highlighted here in green) is excluded from the training set in select experiments.

form is chosen as the exchange−correlation functional. The kinetic energy cutoff of the plane-wave function is set as the default maximum energy cutoff for each material. A Monkhorst−Pack[26] k-point grid of 0.4 2π/Å is used to sample the first Brillouin zone. The convergent criterion for the total energy difference between two successive self-consistency steps is 10^{-5} eV, and all the geometries are fully relaxed until the maximum force acting on each atom is less than 0.01 eV/Å. The elastic constants are calculated from the strain−stress relationship. According to the Voigt−Reuss−Hill (VHR) theory,[27] the corresponding elastic properties, such as the bulk modulus B and shear modulus G, can be evaluated from the elastic constants. To obtain the Grüneisen parameter, we calculate the change in the elastic properties with volume by changing the volume from −1.5 to 1.5% (5 points in total).[28]

Obtaining DFT-computed descriptors for new materials is simple. Some descriptors can be obtained from the crystal structure and chemical components, such as V, M, n, n_p, and ρ. Other descriptors are related to elastic properties, so we only need to compute the elastic constants for new cases using DFT.[28] Bulk modulus B and shear modulus G can be evaluated from the elastic constants according to the VHR theory. Young modulus G and Poisson ration ν are in connection with B and G, as $E = \frac{9BG}{3B+G}$, $\nu = \frac{3B-2G}{6B+2G}$. B' and G' are the derivatives with respect to volume. They can be obtained by changing the volume from −1.5 to 1.5% with an interval of 0.5% (5 points in total including equilibrium structure, i.e., 0% strain). The sound velocities, Debye temperature, and Grüneisen parameters are determined by the elastic modulus with equations listed below

$$\nu_L = \sqrt{\frac{B + 4/3G}{\rho}}, \quad \nu_s = \sqrt{\frac{G}{\rho}}, \quad \nu_a = \left[\frac{1}{3}\left(\frac{1}{\nu_L{}^3} + \frac{2}{\nu_s{}^3}\right)\right]^{-1/3}$$

$$\Theta_D = \frac{h}{k_B}\left[\frac{3m}{4\pi}\right]^{1/3} \nu_a n^{-1/3}$$

$$\gamma_L = -\frac{1}{2}\frac{V}{B + \frac{4G}{3}}\frac{\delta\left(B + \frac{4G}{3}\right)}{\delta V} - \frac{1}{6}$$

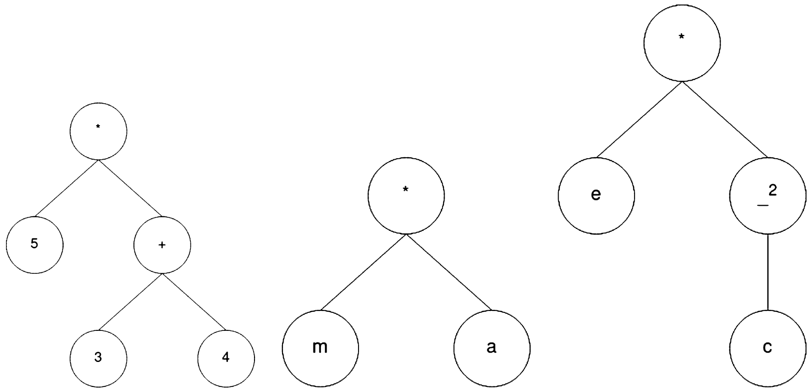
$$\gamma_s = -\frac{1}{2}\frac{V}{G}\frac{\delta G}{\delta V} - \frac{1}{6}$$

$$\gamma_a = \sqrt{\frac{\gamma_L{}^2 + 2\gamma_s{}^2}{3}}$$

2.2. Preprocessing.

The space group descriptor is provided as a categorical value to both the MLP and the RFR models. These models do not output a function; therefore, they are able to process the materials differently based on their space group. For the MLP model, the space group descriptor is converted into binary encoding to be able to process the categorical value properly. In addition to this, the fields shown in Table 1 are scaled using min−max scaling to restrict all variables to a minimum and maximum of 0 and 1. The RF regression model requires that the space group descriptor be converted to ordinal values that represent the various space groups. The dataset needs no preprocessing modification for the SR model aside from the removal of the space group descriptor.

The architectures for the various models are described below. Barring the SR models, there is only one architecture used to create each model.



(a) $5 * (3 + 4)$ (b) $F = ma$ (c) $E = mc^2$

Figure 2. Examples of function trees to be represented in the SR algorithm. The algorithm creates trees similar to the ones above, using the descriptors in the dataset as variables.

https://dx.doi.org/10.1021/acs.jpca.0c08103

2.3. Symbolic Regression.

SR is a form of regression that uses mathematical operators as building blocks to intelligently create formulae, with two objectives: minimizing the prediction error and maximizing the simplicity of the formulae produced. To accomplish this, it uses the concept of the Pareto Frontier[29] to optimize both attributes simultaneously. Through producing simple formulae, SR substantially decreases the likelihood of overfitting to latent trends in the dataset that do not generalize. This is particularly applicable to the fields of physics and material science, as most of the physical laws, when expressed as equations, are relatively mathematically simple. Examples include $F = ma$ and $E = mc^2$.

Our methodology for creating these formulae is through the GP approach.[30] In GP, formulae are represented as unique function trees (see Figure 2 e.g.) with operators, input variables/descriptors, and constants as nodes in each individual tree. They are evaluated for their performance on the dataset and their simplicity using the Pareto frontier. A fitness value is generated for each model. The models are then compared to each other by fitness rankings, with the fitter models having higher probability to proceed onto the next stage of the evolution process: crossover and mutation. The criteria used to determine this fitness score are formula error, complexity, and age. Through the usage of the Pareto Frontier, the algorithm is able to optimize for all three of these criteria simultaneously. The formula error criterion exists to select models that are more accurate. The formula complexity criterion balances out the error criterion and ensures that incredibly complex formulae are not chosen over simpler formulae that have slightly worse performance. This is because very complex formulae may evolve to match the samples in the training set but do not perform well in wider chemical space. Thus, simpler formulae have a better chance of generalizing to diverse chemical space. Finally, the algorithm uses an age criterion to prioritize newer formulae over incredibly old formulae. This is because there are multiple formulae that can approximate κ_L. If we allow the algorithm to favor a specific formula too early in the process, it will only produce formulae that resemble this first formula. This is referred to as a local minimum in the cost function, and we wish to avoid these in favor of finding the absolute minimum.[31]

Subtrees are randomly selected from two partner trees, and offspring in the form of permutations of the parents are created through this crossover stage. In this way, we use natural selection to select successful traits from parents and pass them down to offspring. From here, the genetic process is repeated for a set number of generations, and the most successful formula is returned. GP provides a method for allowing beneficial traits and terms to remain in the function while simultaneously discarding unhelpful terms from the equation. It does not guarantee a perfect solution, but rather through exploring several partial solutions, it is able to intelligently combine them together to create a unified formula for approximating a function that lies underneath a dataset.

SR has a few disadvantages when compared to other ML approaches, but it has one unique advantage that standard ML is unable to replicate. SR generates a human-readable function in a mathematical notation. This formula can be analyzed to derive physical insight into the processes that drive the subject to behave the way that it does. As aforementioned, SR is not a flawless method. It has several disadvantages, among which are computational inefficiency during training and size of the search space. GP-based SR is notoriously computationally inefficient, drawing much more resources for the training phase than statistics-based ML models require. However, once the formulae are produced, they can be run instantly based on their respective complexity, which the algorithm aims to reduce throughout its process.

The size of the search space is of much more concern when applying SR versus most statistics-based ML algorithms. The search space for an SR algorithm is theoretically infinite as there are infinite formulae that can be produced from the GP functions (GP functions) provided to the algorithm. The odds of the algorithm finding and settling for the formula that perfectly maps the provided fields to the desired output are low. In order to offset this large search space, we restrict the height that the function trees are permitted to obtain. This places a finite capacity on the amount of formulae that can be generated while simultaneously ensuring that the formulae we generate remain below a maximum complexity threshold. As aforementioned, physical formulae are mathematically simple, so it is a safe way to prune the search space. To narrow down the search space even further, we restrict the function set that the algorithm is allowed to use. The function sets for these two models are described in Table 2 below. The number of functions provided to the model has a direct correlation to the size of the search space; therefore, by limiting the GP function set, the dimensionality of the problem is reduced, and the likelihood of convergence is increased.

Table 2. Configurations of Two SR Model Architectures

GP1	GP2
×, ÷, $f^{-1}(x)$, random constants on a Gaussian distribution with $\mu = 0$ and $\sigma = 10$	÷, $f^{-1}(x)$, $\ln(\|x\|)$, e^x, x^2, x^3, $\sqrt[2]{(x)}$, $\sqrt[3]{(x)}$, $\sin(x)$, $\cos(x)$, $\tan(x)$, random constants on a Gaussian distribution with $\mu = 0$ and $\sigma = 10$
500 generations	500 generations
2000 population size	2000 population size
7 max height	10 max height
30% mutation probability	30% mutation probability
70% crossover probability	70% crossover probability

This experiment explores two methodologies for calculating κ_L through SR, as described in Table 2. In addition, it uses a third methodology to prove the validity of the SR algorithm for this dataset. The implementation for the SR model was provided by the FastSR library.[32] We selected a GP SR implementation over alternatives, such as SISSO,[33] because of the promise of the aforementioned Pareto Frontier multi-objective optimization.

2.4. Verifying Effectiveness of SR.

In order to provide a benchmark for the validity of our SR algorithm and to demonstrate its ability to learn from a dataset, we created a separate experiment in which an SR model is allowed to train from the Slack predictions for the dataset provided. We provided the SR model with the V, M, θ_D, γ_a, n, n_p, A, and T variables and let it view the Slack model calculated κ_L in order to learn. The goal of this experiment was to demonstrate the learning capacity of our SR methodology by allowing it to train on the Slack predictions and to see how closely it can approximate the Slack formula through exposure to the variables that the Slack formula uses.

The model was permitted to use the following GP functions

$$\times,\ \div\ ,\ f^{-1}(x),\ \ln(|x|),\ e^x,\ \sqrt[3]{(x)},\ x^3,\ \sqrt[2]{(x)},\ x^2$$

Table 3. 10-Fold CV Performance of SR Models, Other ML Models, and the Slack Model[a]

	GP1	GP2	MLP	RFR	Slack	Best
formula	$\frac{G}{H \cdot n_p}$	$\frac{\sqrt[6]{V \cdot n\sqrt{n}}}{\sqrt[12]{\cos(\cos(E))}}$			$A \cdot \frac{(\theta e)^3 M \sqrt[3]{V n_p}}{n^{4/3} T {\gamma_a}^2}$	
RMSE	15.914	16.184	11.816	**5.870**	16.349	RFR
R^2	0.368	0.346	0.651	**0.914**	−1.206	RFR

[a] Bold values correspond to the best ML/SR models.

In addition, randomly generated constants following a Gaussian distribution with $\mu = 0$ and $\sigma = 10$ are also provided to evolve the coefficients in the formulae. The algorithm created 1000 generations of 1500 formulae. The model is limited to producing formula trees with a maximum height of 7, and 5 of the 1500 functions introduced with each generation were generated completely randomly, in order to prevent the model from fixating on a local minimum in the cost function gradient. Ultimately, our SR algorithm evolved a formula (see Formula 3) that is extremely close to that of the Slack formula, its target. The evolved formula achieves an RMSE of 5.296 and R_2 of 0.946. The parity plot of the predicted κ_L versus Slack model values is shown in Figure 3. These results reflect the ability of the evolved SR formula to map inputs to their predicted κ_L Slack values.

$$\kappa_L = \sqrt[3]{\frac{\theta^{37/9}\, e^{74/9} M^2 \gamma}{n^2 T^7}} \tag{3}$$

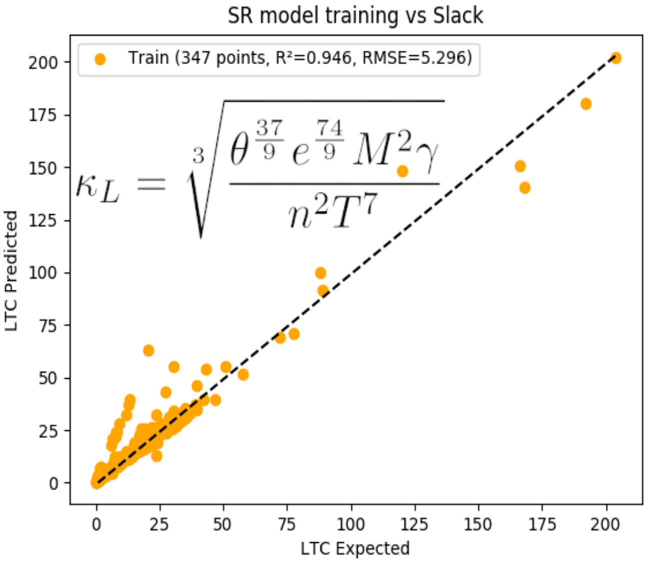


Figure 3. Parity plot for the evolved Formula 3.

Formula 3 is the simplified form of the evolved formula, which boasts a very high R^2 score of 0.946, meaning that it very closely mirrors the Slack equation. Interestingly, Formula 3 does not use the A, V, and n_p variables. It is intriguing that it is able to obtain an effective approximation without making use of three of the variables from the equation it is trying to replicate. Despite this, it still obtains an effective approximation. Another interesting observation is that Formula 3 places the γ_a variable in the numerator rather than the denominator and changes the exponent from 2 to 1/3. This creates a relationship where the κ_L and the γ_a values have a partial direct correlation. Typically, κ_L is inversely proportional to the square of γ. However, the relationship between κ_L and γ is complicated. Rigorously speaking, there is no analytical formula or relationship between κ_L and γ. Accurate assessment of γ is computationally expensive, so γ is incorporated into A, A_1, and A_2, which are fitted parameters and related to Grüneisen parameter γ[18] in Formulae 1 and 2. Because of the difference of dataset, the lack of data points, and the inaccuracy of descriptors obtained from DFT, the prediction models could be different. For example, in the ML models proposed by Juneja et al.,[34] the γ variable has an exponent of 1/4.

The θ_D variable in the numerator of Formula 1 has an exponent of 3, yet the same variable in Formula 3, through some algebraic manipulation, has an exponent of $1 + \frac{10}{27}$, meaning that Formula 3 places less importance on the Debye temperature than the original Slack equation. These changes in scaling could be a result of the model compensating for the missing variables, thus demonstrating the plastic and adaptive nature of the SR algorithm. The fact that it is able to reproduce the Slack equation with an R^2 value of 0.946 means that it has the potential to regress a formula with a comparable coefficient of determination with the actual κ_L values set as the supervised learning set.

2.5. MLP Neural Network.

Neural networks are mathematical models that take in a predefined number of inputs and convert them through multiple layers of linear or nonlinear transformation to generate a predefined number of outputs.[35,36] It is well-known that deep neural networks are excellent at learning nonlinear relationships,[36] but deep learning approaches such as the MLP require vast amounts of data to effectively learn trends and relationships. In addition, MLP models form a black box system that, while accurate, is unable to provide scientists with insight into how the model is able to map the input variables to their expected output variables; they are a tool that can be used, but their processes for reaching their solutions cannot be understood easily despite recent efforts to create explainable deep neural network models.[37]

In order to ensure that our model is able to adequately learn from the dataset, we allowed the model to train over 30 epochs for each step in the fivefold CV process. To offset any overfitting which may manifest as a result of this process, we use random dropout to address the issue.[38] The MLP model, as depicted in Figure 4, makes use of 5 hidden layers with 1024 neurons in each layer and a 20% dropout between otherwise densely connected layers. Rectified linear unit is the activation function for all layers leading up to the final layer, which uses linear activation. The network was trained with MAE as the loss function and makes use of the Adam optimizer.[39]

2.6. RF Regressor.

RF is an ensemble ML algorithm that takes advantage of a predefined number of decision trees.[40] Our RFR implementation uses a standard RF model as

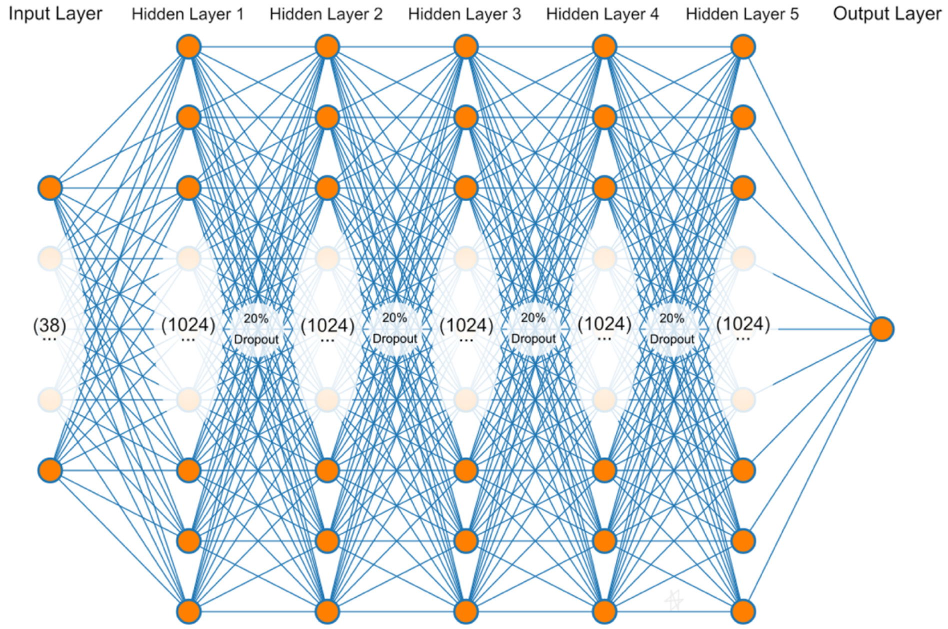


Figure 4. Architecture of the MLP model.

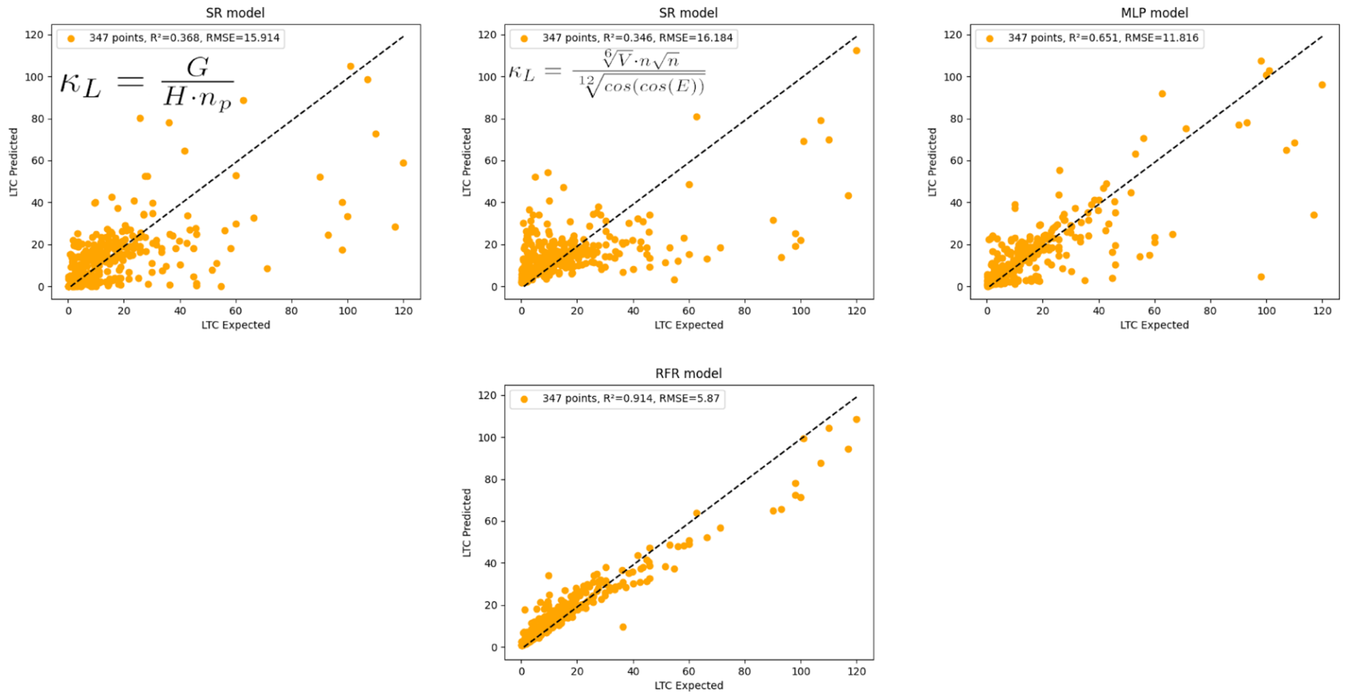


Figure 5. Parity plots for the 10-fold CV experiments of GP1, GP2, MLP, and RFR.

provided by the scikit-learn[41] Python package. The number of estimators is set as 100 and the loss function is set to MAE. By minimizing MAE rather than RMSE, the RFR model aims to provide a smooth prediction over all values rather than overpunishing high residuals in κ_L predictions.

3. EXPERIMENTS AND RESULTS

3.1. Training Process. We conduct two types of experiments to compare the SR and ML models including the standard CV tests and forward CV extrapolation performance tests.

For CV experiments, during the training process, the data points are split into 10 equal subsets, and then 10-fold CV is performed.[42,43] The MLP was permitted to train for 30 epochs over the training set during each training interval of the CV process.

For extrapolation test experiments, all models are trained on 80% of the dataset and then evaluated on a block of the remaining 20% of the dataset in a process known as extrapolation testing.[15] We also implement fivefold CV on the training set to reduce the chance of overfitting.[42,43] For example, one of the extrapolation tests (depicted in Table 7 and Figure 8) sorts the materials in the order of ascending κ_L and allows the model to train on the middle 80% of the data points. The bottom 10% and the top 10% are withheld from the model during the training phase and retained for the

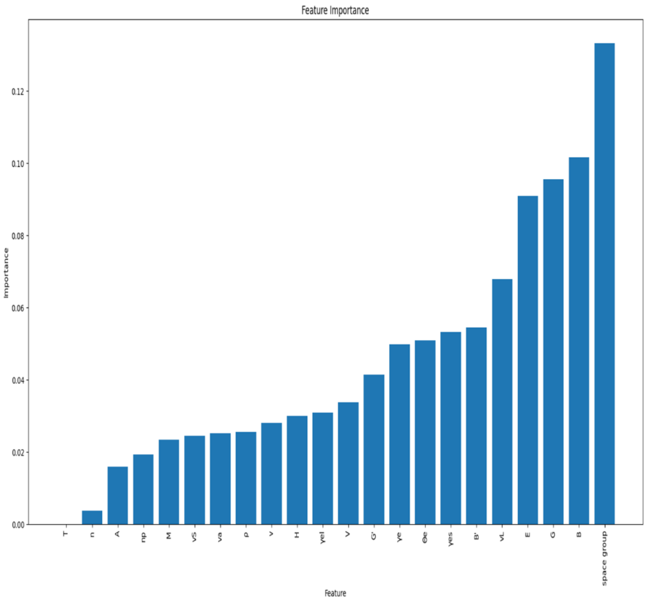


Figure 6. Bar graph indicating the importance of the various features utilized by the RFR model in Table 3. A higher *Y*-value indicates that the RFR model paid more attention to this field, whereas a lower *Y*-value indicates that it was not very important to the calculation.

validation set. After the model has been trained, the model is tested on the validation set. If the model is able to perform similarly on both the training and validation sets, it is understood that the model has learned an underlying relationship between the input variables and κ_L. Because most of the test samples are not neighboring training samples in such tests, it is guaranteed that this relationship is not purely based on the samples' proximity. The relationship learned can be used to predict κ_L values of any sample and thus must reflect the physical law that underlies κ_L approximation.

3.2. Random CV Results. As shown in Figure 5, the SR models and the MLP and RFR models have all achieved good CV prediction performance with R^2 scores of 0.368, 0.346, 0.651, and 0.914. In this evaluation approach, the samples are randomly shuffled and split into $\kappa = 10$ folds. Thus, the test samples also have a chance to find similar neighboring samples, thus good prediction performance. Overall, the RFR has achieved the best performance with an R^2 of 0.914. Compared to its low extrapolation performance as shown in the next section, the standard CV is the best for estimating interpolation performance.

Using the RFR model, we have calculated the Gini importance[44] in order to get an idea of which features are most important to the RFR model's calculations. Because this model achieved an R^2 score of 0.914, it is a good metric of which features have the most impact on κ_L calculation. As shown by Figure 6, the space group field had the most impact on the RFR model's calculation of κ_L.

3.3. Extrapolation Testing Results. Tables 4–7 show the results from the four extrapolation testing sets. The *Best* column indicates the model that had the best performance on that set. All of the new formulae found in this section are analyzed in greater detail in Section 3.6. The formulae displayed in the table have been simplified from the original forms created by the computational models and thus may contain operators that exist outside of their associated function sets, as defined in Table 2. These new operators are the result of combining operators used by the models. For example, $n_p \times n_p$ is simplified to n_p^2. Some of the formulae (particularly those produced by GP2) appear very strange; we believe that this is due to the model attempting to approximate an expression but misappropriating GP functions. GP1 does not suffer from this problem, which has led us to believe that it stems from the large number of GP functions that the GP2 model was provided with. Although these formulae may appear curious, we must keep in mind that they are the result of the model creating approximations of the true function using the GP functions provided. This is similar to how the Taylor series can approximate other functions using polynomials, which perform favorably despite often not resembling their target function when viewed symbolically.

3.3.1. Performance Comparison of SR and ML Models to the Slack Model. First, Table 4 shows the prediction performance of the algorithms when trained with top 80% samples and tested on bottom 20% samples. Over the training sets, the MLP model achieves the best performance with an RMSE of 18.792 and an R^2 of 0.188. On the testing set, the MLP model outperforms the GP1 model in RMSE (4.166 vs 5.089), although not in R^2 (−45.97 vs −38.381), indicating that the neural network's predictions have more variance than the function that GP1 produces. It should be noted that the coefficient of determination R^2 can become negative when evaluated over test sets which are not included in the training set. GP2 is able to obtain 0.938 less RMSE than the Slack model on the training set, but 0.831 more error on the testing set, which shows that their model learned ungeneralizable trends in the subset of the data that it was shown. Strangely, although GP2 is evolved with more evaluations than the GP1 model and had access to more GP functions than GP1, the GP1 model achieved better metrics on the testing set than the GP2 model did. We attribute this incongruity to the larger search space that the GP2 model must navigate because of its larger pool of GP functions.

We observe similar performance advantages of SR models compared to MLP, RFR, and Slack models in Table 5, which shows the performance of models when trained with top 40% and bottom 40% samples and tested on the middle 20% samples. The MLP model is able to outperform all of the other models when evaluated against the set of data that it was trained upon; however, the GP1 model is superior on the testing set. This demonstrates that the GP1 model was able to learn trends from extremely poor and extremely successful thermal conductors and accurately apply those trends to gain insight on the materials that lie in between those extremes. Interestingly, the GP1 model and RFR model are the only models that performed worse than the Slack model on the training set—both GP2 and MLP were able to outperform the Slack model on the training set. The GP2 model is able to outperform the Slack–Berman model across all metrics on all subsets of the dataset. It has a spectacularly better performance than the Slack model on the testing set and performs better on the training set as well.

For the other two extrapolation experiments shown in Tables 6 and 7, the results are a little bit different. Table 6 shows the results of models trained with bottom 80% samples and tested on the top 20% samples while Table 7 shows the results of models trained with middle 80% samples and tested on the two-end 20% samples. In Table 6, the MLP model performs extraordinarily well. It is able to outperform all other SR/ML models across all of the testing metrics and obtains

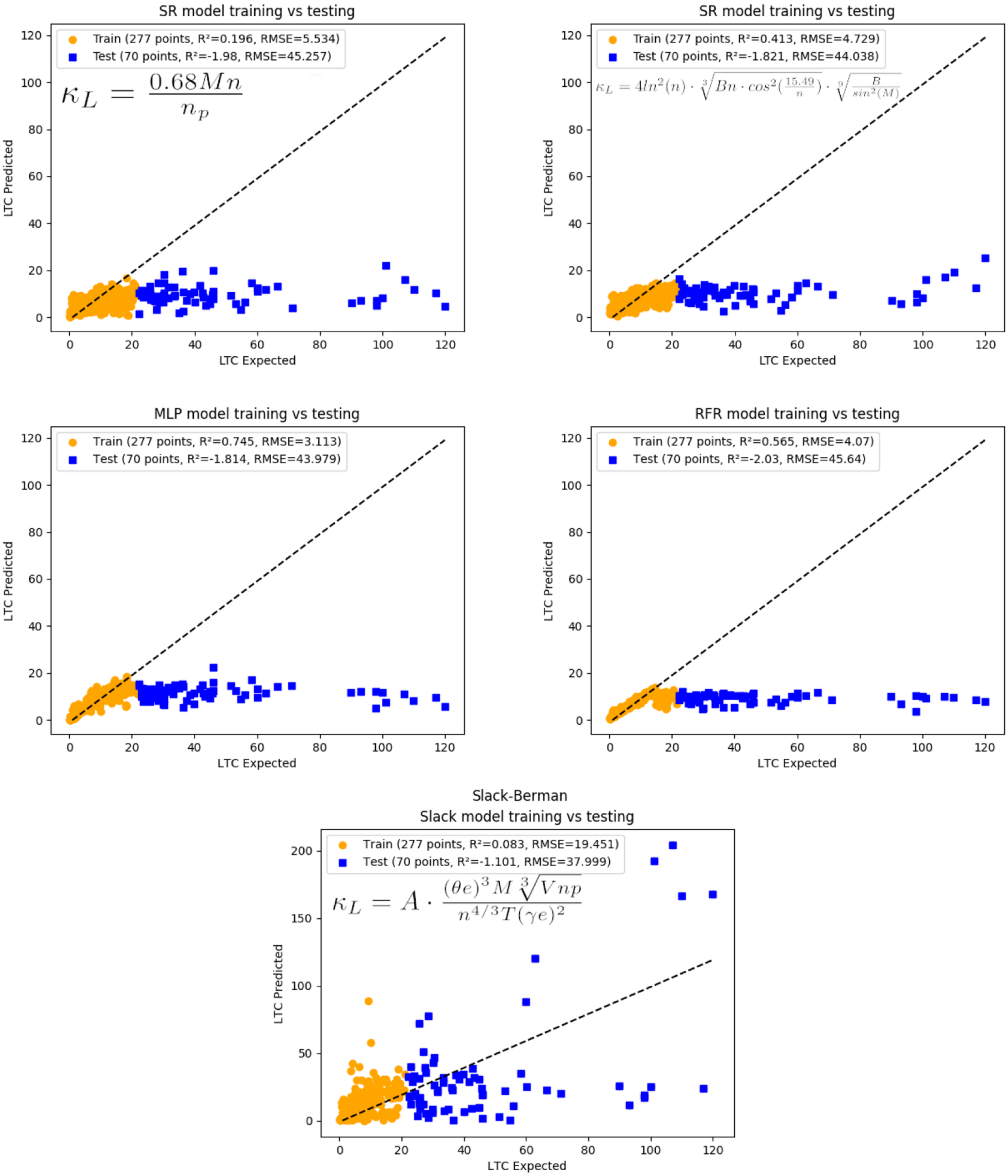


Figure 7. Parity plots for the GP1, GP2, MLP, RFR, and Slack–Berman models on the top 20% extrapolation testing set.

6.25× lower RMSE than the Slack model on the training set. Unfortunately, it underperforms on the testing set, demonstrating less aptitude than the Slack model for predicting materials at the upper end of the κ_L spectrum. In Table 6, the best SR model GP2 achieves an RMSE of 44.038 and an R^2 of −1.821 over the test set which is worse than the RMSE of 37.999 and an R^2 of −1.101 of the Slack model. In Table 7, the GP2 model outperforms all of the other ML/SR models on the testing set, although it still underperforms when compared to the Slack model. Further attention to Tables 6 and 7 reveals that when trained on the middle 80%, the GP1 model produces a formula with a 139.48% increase in the R^2 score as compared to its score in Table 6. Similarly, GP2 sees a 156.36% increase in its own R^2 score. As expected, the results

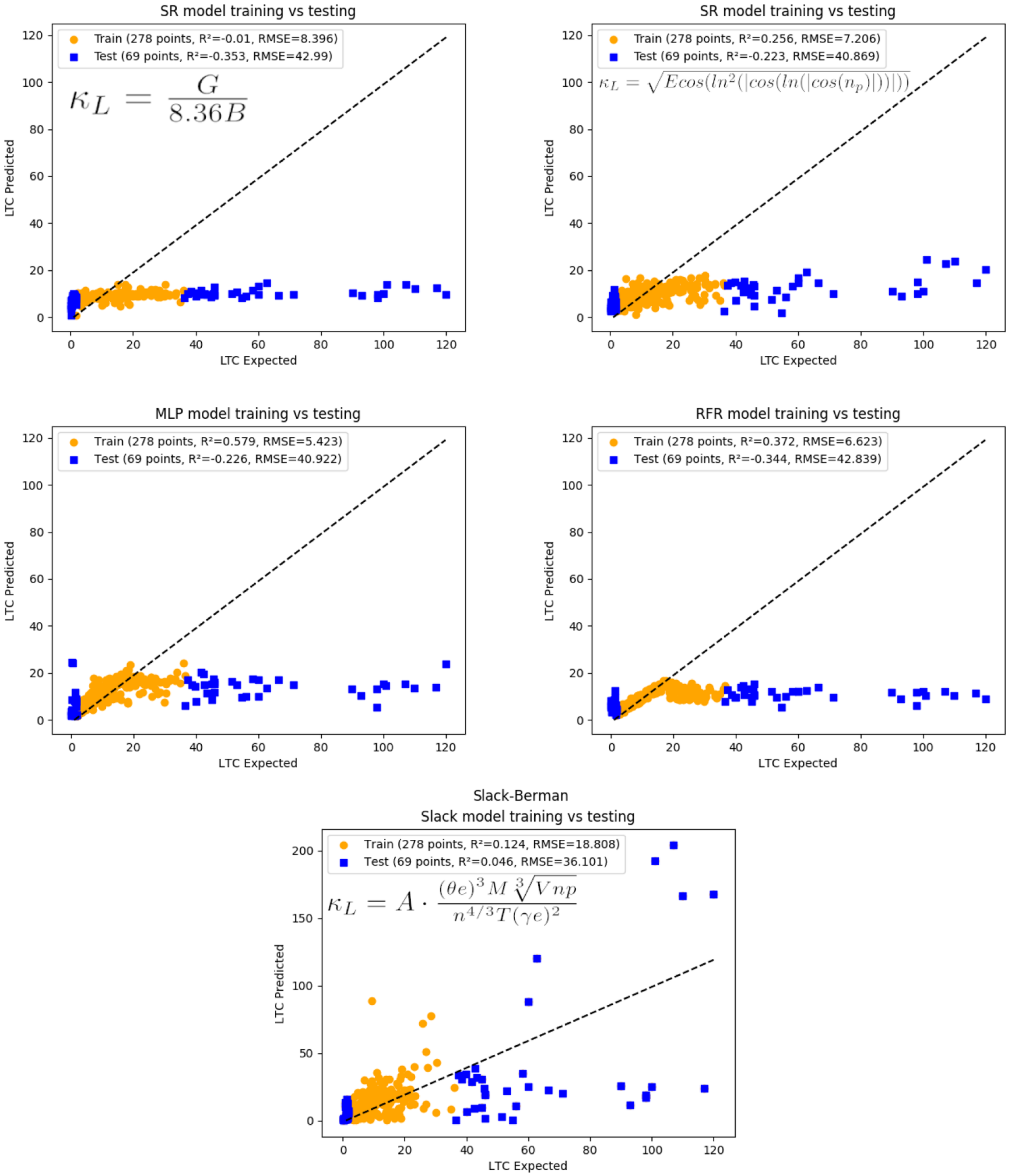


Figure 8. Parity plots for the GP1, GP2, MLP, RFR, and Slack−Berman models on the top 10% and bottom 10% extrapolation testing set.

demonstrate that training the model on a diverse set of data points yields increased extrapolative power. Naturally it should follow the fact that the most diverse training set should provide the most extrapolative potential. Table 5 shows that this is true.

As noted above, training models on a diverse set of data provides the most extrapolative potential as opposed to alternative methods. Naturally, it follows that training the model on the bottom 40% of the dataset and top 40% of the dataset would yield the most accurate formulae as the SR models would be exposed to examples of both materials with high and low κ_L. This is supported by Table 5, in which both GP2 and GP1 yield formulae that outperform the Slack−Berman equation. On the training set, GP1 and GP2 perform comparably to the Slack model, with GP1 estimating κ_L with

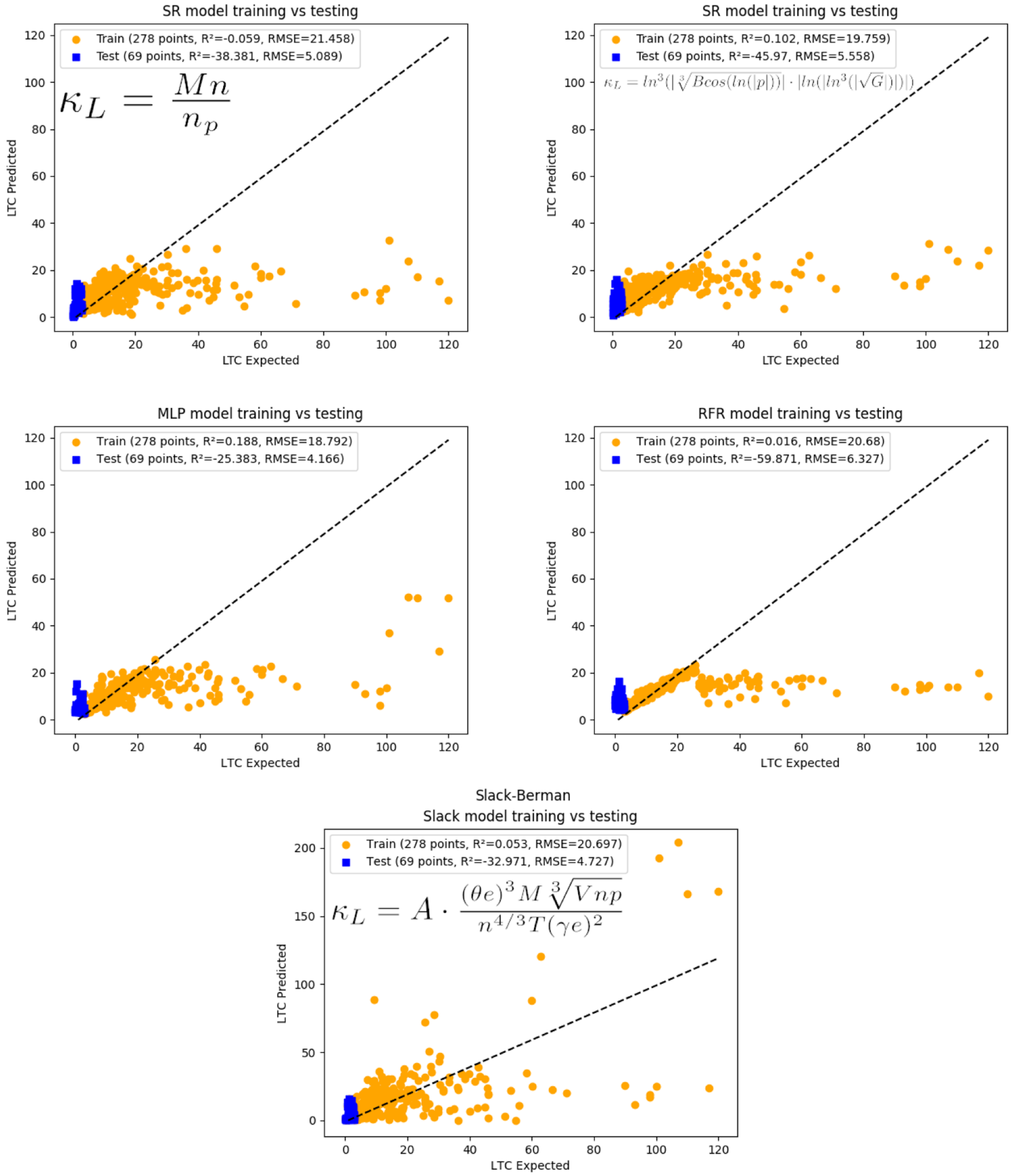


Figure 9. Parity plots for the GP1, GP2, MLP, RFR, and Slack−Berman models on the bottom 20% extrapolation testing set.

0.901 more RMSE and GP2 estimating κ_L with 0.994 less RMSE. However, on the testing sets, GP1 and GP2 demonstrate that they are significantly more accurate. GP1 has 9.241 RMSE less than the Slack model on the testing set, and GP2 achieves an RMSE of 5.825, 8.204 less than the Slack model on the same set (14.029). The two formulae produced by GP1 and GP2 perform similarly to the Slack model on the training set but are able to predict the median 20% of materials with 2.93× and 2.4× less error than the Slack model, despite the fact that they have never seen materials with κ_L in that range before. This successful prediction proves that the SR models are not overfitting to latent trends in their training sets but have uncovered relationships that govern the calculation of κ_L. Formula 4 represents the formula generated by GP1 from

Table 4. Results after Training the Models on the Top 80% of the Samples with the Highest κ_L Values and Testing on the Bottom 20% Samples[a]

	GP1	GP2	MLP	RFR	Slack	Best
formula	Mn/n_p	$\ln^3(\lvert\sqrt[3]{B\cos(\ln(\lvert p\rvert))}\rvert\cdot\lvert\ln(\lvert\ln^3(\lvert\sqrt{G}\rvert)\rvert)\rvert)$			$A\cdot\frac{(\theta e)^3 M\sqrt[3]{Vn_p}}{n^{4/3}T{\gamma_a}^2}$	
training RMSE	21.458	19.759	**18.792**	20.680	20.697	MLP
testing RMSE	5.089	5.558	**4.166**	6.327	4.727	MLP
training R^2	−0.059	0.102	**0.188**	0.016	0.053	MLP
testing R^2	**−38.381**	−45.97	−45.97	−59.871	−32.971	GP1

[a] Bold values correspond to the best ML/SR models.

Table 5. Results after Training the Models on the Top and Bottom 80% of Samples with Lowest 40% and Highest 40% κ_L Values and Testing on the Middle Samples[a]

	GP1	GP2	MLP	RFR	Slack	Best
formula	$0.31B/n_p$	$B^{(1/2)}{n_p}^{1/9}\cos(\cos(\sin(\cos(V))))\cos^2\left(\frac{1}{\tan^3(e^n)}\right)$			$A\cdot\frac{(\theta e)^3 M\sqrt[3]{Vn_p}}{n^{4/3}T{\gamma_a}^2}$	
training RMSE	20.293	18.398	**17.332**	20.941	19.392	MLP
testing RMSE	**4.788**	5.825	4.992	5.457	14.029	GP1
training R^2	0.159	0.309	**0.387**	0.105	0.036	MLP
testing R^2	**−6.718**	−10.424	−7.391	−9.027	−65.266	GP1

[a] Bold values correspond to the best ML/SR models.

Table 6. Results after Training the Models on the 80% of the Samples with the Lowest κ_L Values and Testing on the Top 20% Samples[a]

	GP1	GP2	MLP	RFR	Slack	Best
formula	$0.68Mn/n_p$	$4\ln^2(n)\cdot\sqrt[3]{Bn\cdot\cos^2\left(\frac{15.49}{n}\right)}\cdot\sqrt[9]{\frac{B}{\sin^2(M)}}$			$A\cdot\frac{(\theta e)^3 M\sqrt[3]{Vn_p}}{n^{4/3}T{\gamma_a}^2}$	
training RMSE	5.534	4.729	**3.113**	4.070	19.451	MLP
testing RMSE	45.257	44.038	**43.979**	45.640	37.999	MLP
training R^2	0.196	0.413	**0.745**	0.565	0.083	MLP
testing R^2	−1.98	−1.821	**−1.814**	−2.030	−1.101	MLP

[a] Bold values correspond to the best ML/SR models.

Table 7. Results after Training the Models on the 80% Samples with Middle κ_L Values and Testing on the Top and Bottom 10% of Samples[a]

	GP1	GP2	MLP	RFR	Slack	Best
formula	$G/8.36B$	$\sqrt{E\cos(\ln^2(\lvert\cos(\ln(\lvert\cos(n_p)\rvert))\rvert))}$			$A\cdot\frac{(\theta e)^3 M\sqrt[3]{Vn_p}}{n^{4/3}T{\gamma_a}^2}$	
training RMSE	8.396	7.206	**5.423**	6.623	18.808	MLP
testing RMSE	42.99	**40.869**	40.922	42.839	36.101	GP2
training R^2	−0.01	0.256	**0.579**	0.372	0.124	MLP
testing R^2	−0.353	**−0.223**	−0.226	−0.334	0.046	GP2

[a] Bold values correspond to the best ML/SR models.

this training set, and Formula 5 represents that generated by GP2. We discuss these formulae further in Section 3.6, but for convenience, we provide them here.

Now, there is one remaining question: why do the SR models work better when trained with the top 80% and tested on the bottom 20% compared to when they are trained with the bottom 80% and tested on the top 20%? After close inspection of the sample distribution in Figure 1, it seems that this is caused by the extremely sparse amount of samples in the high-κ_L area compared to the dense amount of samples in the bottom κ_L area. As a result, whenever the test set includes the top κ_L area, the extrapolation performance will be very low. This result confirms the importance of training ML and SR models with balanced, diverse data samples.

$$\kappa_L = \frac{0.31B}{n_p} \tag{4}$$

$$\kappa_L = B^{(1/2)}{n_p}^{1/9}\cos(\cos(\sin(\cos(V))))\cos^2\left(\frac{1}{\tan^3(e^n)}\right) \tag{5}$$

3.3.2. Performance Comparison of SR, RFR, and MLP.

Comparing the SR and ML model performance against each

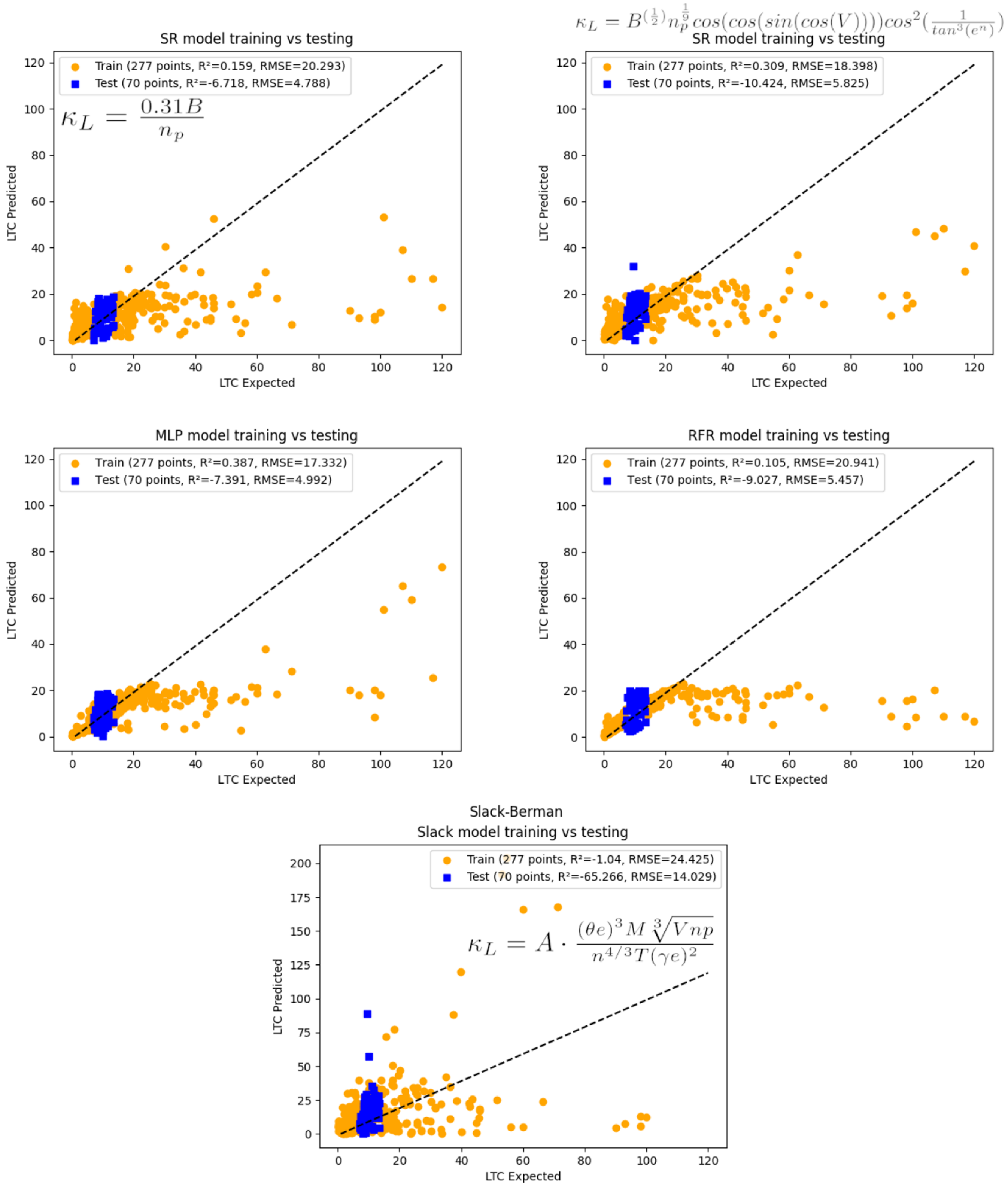


Figure 10. Parity plots for the GP1, GP2, MLP, RFR, and Slack−Berman models on the middle 20% extrapolation testing set.

other on the datasets reveals a few notable results. Although the MLP model is very effective at learning from the data it is shown, it does not have the same extrapolative potential that the SR models have. MLP outperforms all other models on the training sets when evaluated by both RMSE and R^2, as shown in Tables 4−7. However, it is not always able to outperform the GP models on the testing sets. The MLP model's predictions on the validation sets are close to that of the SR models, with the largest difference in RMSE being 2.068, as shown in Table 7. The fact that it is unable to consistently match the SR models' performance on the testing sets yet outperforms them on the training sets demonstrating that although the neural network is excellent at creating a model that accurately predicts materials in the range it has seen before, it does not always transfer this knowledge to materials outside of this range.

The RFR model does not perform better than the MLP model for any metric on any set of materials from the dataset. However, it does obtain lower RMSE and higher R^2 values than the SR models on the training sets for Tables 6 and 7. Despite outperforming the SR model on the training sets, the MLP model is unable to compare with the SR models on any of the validation sets except for Tables 5 and 7, where it outperforms one of the SR models but is surpassed by the other SR model. For example, in Table 5, the RFR model outperforms the GP2 formula's RMSE by 0.368 but is worse than GP1's formula by 0.669. None of the models were able to perform better than the Slack model on the extrapolation sets depicted in Tables 6 and 7. However, this is not to say that the Slack formula is superior for calculating the κ_L of materials in those validation sets; it simply means that the models needed data points from those sets in order to learn the relationships for them. The models demonstrated their efficacy for κ_L approximation on those materials through the RMSE reported in Tables 4 and 5, where they demonstrated comparable performance to the Slack formula.

3.3.3. Performance Comparison of GP1 and GP2.

As shown in Table 2, we evaluated two SR algorithms to evolve SR models: GP1 and GP2, where GP1 corresponds to a simpler function set with a max tree height of 7, leading to simpler models. On the other hand, the GP2 model is trained with a more complex function set with a tree height of 10, leading to more complex models.

All the GP1 and GP2 performance results with four extrapolation experiments are shown in Tables 4–7. We find that the GP1 model outperforms the GP2 model when the top portion of the dataset is included in the testing set (Tables 4 and 5). This can be noted by observing Table 4, in which GP1's RMSE (5.089) is 0.469 less than GP2's RMSE of 5.558. However, when the top section of the dataset is excluded from the training set (as in Tables 6 and 7), the GP2 model outperforms the GP1 model. This can be observed by comparing a GP1's RMSE score of 42.99 to a GP2's RMSE of 40.869, as shown in Table 7.

3.4. Parity Plot Analysis.

To further understand why the SR and ML models have unexpected low extrapolation prediction performance, we create a set of parity plots (Figures 7–10) for all the four extrapolation experiments and aim to figure how the sample distribution affects the prediction performance of ML and SR models. In all of the plots, the orange points represent training samples while the blue points are test samples. However, for the Slack model, both colors represent testing points. This is because the Slack model is an empirical model and thus does not require training.

First, we find that compared to the random CV performance results (Figure 5), the extrapolation prediction performances of all SR and ML models are unexpectedly low, consistent with our previous observations[15] along with other analysis[45] on out-of-distribution generalization issues.

Second, across all the parity plots, there is a clear propensity for the prediction models to underestimate κ_L (most of the samples are below the diagonal line). This is not an unexpected development, as the dataset contains many more materials on the lower spectrum of κ_L materials than the upper bound. As Figure 1 demonstrates, its distribution is positively skewed. The models that were trained on materials with lower κ_L values often underestimate the values of their testing sets, as shown in Figure 7. The models trained on the upper side of the material κ_L spectrum tend to generate overestimates, as demonstrated in Figure 9. In Figure 8, the models both overapproximate the lower κ_L materials and underestimate the higher κ_L materials' values, as the models shown in this figure were trained on the middle 80% of the dataset. Finally, Figure 10 shows a more even balance, with variance both above and below the $y = x$ line. This is due to the fact that the models were provided a diverse training set of both high and low κ_L materials, as previously noted in Section 3.3. They still show a tendency to underpredict the values on the upper bound, with the SR models demonstrating a slightly more standard yet still skewed variance.

The parity plots (Figures 7–10) for all extrapolation sets indicate that the models behaved largely as expected on their training sets; on some materials, the models overestimated κ_L, and on others, they underestimated κ_L. The RFR model's parity plots have an interesting spread on the training sets. Most easily seen in Figure 9, the RFR model predicts the lower end of its training sets with relatively low error, but then abruptly begins predicting nearly the same κ_L for all of its materials with some variation. This variation is the lowest as shown in Figure 7 and the highest as shown in Figure 10. The point at which the RFR model experiences its estimation accuracy falloff occurs at a logical point in each of its parity plots. In Figure 7, the jump in error occurs relatively early in the plot, whereas in Figures 9 and 10, it occurs later. This is because Figure 7 contains materials of low κ_L, so the error spikes when the κ_L increases. This same spike occurs later in the other figures because they include member nodes of higher κ_L values for the model to learn from. The RFR models' parity plot testing sets continuous trends identified by the models for the higher values in their training sets, which indicates that the models have found similarities in the fields of the two subsets. This sudden spike in performance is not an unexpected development as the RFR model is decision tree-based.[15]

3.5. Formula Comparison.

Comparing the formulae generated by the SR models yields some interesting insight, particularly when they are compared to the Slack–Berman Formula 1. Both formulae produced by the SR model in the extrapolation experiments (eqs 4 and 5) place the B field in the numerator, indicating that κ_L scales with bulk modulus. This is consistent with current kinetic theory of phonon transport that the inclusion of bulk modulus as a variable in the formula is essential to approximating a material's κ_L.

Significantly, eqs 6 and 7 do not make use of the B field whatsoever. This contradicts the aforementioned kinetic theory of phonon transport. In addition, the only variable that eq 6 has in common with eqs 4 and 5 is n_p. Equations 5 and 7 have some overlap in which they both make use of the V and n variables.

Equation 5 places the n_p variable in the numerator, which corresponds to the Slack formula's usage of the field. However, eq 4 places it in the denominator. This disparity indicates a disagreement between the formulae, where eq 5 assumes that κ_L has a positive correlation with the number of atoms in a primitive cell and eq 4 indicates that they have a negative correlation. We conclude that eq 5's usage of the field is most likely correct, as it boasts a lower RMSE than eq 4 (16.643 vs 18.255).

The set of variables selected by the formulae is perhaps the most interesting result. The Slack formula makes use of 6 variables and 2 constants (A & T), whereas the most accurate formula that our models produced (Formula 7) uses only three variables and achieves a higher accuracy. The two formulae have the n and V variables in common.

Although the models produced a multitude of potential formulae, we have elected only to include those with the least error in the primary section of this work. A selection of other noteworthy formulae have been collected based on their interesting properties and have been included in the Supporting Information paper.

$$\kappa_{\mathrm{L}} = \frac{0.31B}{n_{\mathrm{p}}} \tag{4}$$

$$\kappa_{\mathrm{L}} = B^{(1/2)}{n_{\mathrm{p}}}^{1/9}\cos(\cos(\sin(\cos(V))))\cos^{2}\left(\frac{1}{\tan^{3}(\mathrm{e}^{n})}\right) \tag{5}$$

$$\kappa_{\mathrm{L}} = \frac{G}{H\cdot n_{\mathrm{p}}} \tag{6}$$

$$\kappa_{\mathrm{L}} = \frac{\sqrt[6]{V}\cdot n\sqrt{n}}{\sqrt[12]{\cos(\cos(E))}} \tag{7}$$

3.6. Discussion. All of the ML models reviewed in this study have their own unique advantages and disadvantages to their use. SR is computationally expensive and time-consuming during the training stage, but it leads to formulae that are physically meaningful and have enhanced extrapolative capacity and speed during the prediction stage. RF reduces overfitting and variance through the usage of bagging and ensemble learning. MLP neural networks are able to accurately discover nonlinear relationships from training data, and with a large enough dataset, they are able to use this information to estimate data points that lie outside their training set.

Using RMSE and R^2 as metrics for evaluation, the SR models used in this work were collectively more effective than any other models on the extrapolation validation sets. The MLP model performed comparably on a number of validation sets and outperformed the GP1 and GP2 models on some others, but overall it was less effective on the validation sets. In addition, the SR models provide formulae that can be analyzed to obtain physical insight into the relationships of the variables in the formulae; MLP and RF models cannot provide the same level of insight.

Our SR models produced formulae that are able to calculate κ_{L} with comparable or greater accuracy than the traditional Slack formula (eq 1), all while using less variables to do so. We demonstrate the validity of our SR methodology by showing that it can approximate the Slack formula with an R^2 score of 0.946 (Figure 3). There are a multitude of other sources that have proven the SR algorithm's capacity for discovering physical laws.[46,47] SR provides computers with the ability to discover natural laws from raw data and even provides physical insights. Formulae 4 and 5 successfully reproduce known physical insight that κ_{L} scales positively with bulk modulus. Formulae 4 and 6 reproduce the physical insight that κ_{L} scales negatively with the number of atoms in the primitive unit cell of a material.

In addition to the other models discussed in the paper, we also ran the lastest SR algorithm, the AI Feynman algorithm[48] over our dataset using the implementation in the github repository by Udrescu.[49,50] Initially, the algorithm did not converge to any usable formula because our dataset contained too many input variables. However, even after we restricted the dimensionality of the problem to only the six variables used by the original Slack model and allowed the model to run continuously for nine days, it still did not converge. The AI Feynman algorithm on paper is a very strong candidate for predicting formulae for LTC, as it is an SR algorithm that does not rely on GP. Rather than using an evolutionary algorithm, the AI Feynman algorithm uses neural networks to simplify the data it is provided with before using a brute force algorithm to try all symbolic expressions possible in the order of ascending complexity. The algorithm is very promising and has the capability to exploit the units of variables. Unfortunately, with our currently limited dataset, we were unable to successfully apply it to get better formulae.

There are a few areas in which this study could be improved. First, collating a larger dataset of materials with measured κ_{L} values will enable all three types of models explored in this study to obtain lower error metrics and resolve issues with variance and bias from the models. This could be accomplished using material feature generation toolkits, such as Magpie and Matminer.[51,52] Obtaining a balanced dataset that has a normally distributed range of κ_{L}-ranked materials will permit the models to improve their performance across all categories, especially when predicting materials with high κ_{L}. Outside of changes to the dataset, the SR methodology could be improved. Attributing units and types to the variables in the dataset before feeding them to the SR models will allow for the inclusion of binary operators that require consistent units, such as addition and subtraction. The inclusion of these operators would substantially increase the hypothesis space of the SR models, potentially leading to more accurate models.

Even with these limitations, SR has demonstrated that it can learn from raw experimental data and intelligently produce equations and formulae that can predict unseen values. In this work, we have proven that GP has the capacity to create formulae that are more accurate and more consistent than models that have been derived by physicists for the same task (Formula 1). It can infer relationships that are relevant to materials outside of the range that it was trained on, and it does so with less error than neural networks and RFRs trained on the exact same data.

■ ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acs.jpca.0c08103.

Several formulae that performed well during tests, but whose performance is inadequate in comparison to those mentioned in the main body of this article and formulae and their accompanying RMSE and R^2 metrics for those who wish to view them (PDF)

■ AUTHOR INFORMATION

Corresponding Authors

Ming Hu − *Department of Mechanical Engineering, University of South Carolina, Columbia, South Carolina 29208, United States;* orcid.org/0000-0002-8209-0139; Email: hu@sc.edu

Jianjun Hu − *Department of Computer Science and Engineering, University of South Carolina, Columbia, South Carolina 29201, United States;* orcid.org/0000-0002-8725-6660; Email: jianjunh@cse.sc.edu

Authors

Christian Loftis − *Department of Computer Science and Engineering, University of South Carolina, Columbia, South Carolina 29201, United States*

Kunpeng Yuan − *Department of Mechanical Engineering, University of South Carolina, Columbia, South Carolina 29208, United States; Key Laboratory of Ocean Energy Utilization and Energy Conservation of Ministry of Education, School of Energy and Power Engineering, Dalian University of Technology, Dalian 116024, China*

Yong Zhao − *Department of Computer Science and Engineering, University of South Carolina, Columbia, South Carolina 29201, United States;* orcid.org/0000-0002-6762-266X

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jpca.0c08103

Author Contributions

Conceptualization: J.H. and M.H.; methodology: J.H. and C.L.; software: C.L., J.H., and Y.Z.; validation: C.L., J.H., and M.H.; investigation: C.L., J.H., and M.H.; resources: J.H.; data curation: K.Y. and M.H.; writing−original draft preparation: C.L and J.H.; writing−review and editing: C.L., J.H., Y.Z., K.Y., and M.H.; visualization: C.L.; supervision: J.H.; project administration: J.H.; and funding acquisition: J.H. and M.H.

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

This research was supported, in part, by a grant from the Magellan Scholar program, from the Department of Undergraduate Research at the University of South Carolina, Columbia. Research reported in this publication was also partially supported by the National Science Foundation under grant numbers: 1940099, 1905775, and OIA-1655740 (via SC EPSCoR/IDeA 20-SA05) and by DOE under grant number DE-SC0020272.

■ REFERENCES

(1) Morelli, D. T.; Slack, G. A. High Lattice Thermal Conductivity Solids. In *High Thermal Conductivity Materials*; Shindé, S. L., Goela, J. S., Eds.; Springer New York: New York, NY, 2006; p 44.

(2) Callaway, J. Model for Lattice Thermal Conductivity at Low Temperatures. *Phys. Rev.* **1959**, *113*, 1046−1051.

(3) Chen, L.; Tran, H.; Batra, R.; Kim, C.; Ramprasad, R. Machine Learning Models for the Lattice Thermal Conductivity Prediction of Inorganic Materials. *Comput. Mater. Sci.* **2019**, *170*, 109155.

(4) Ma, J.; Li, W.; Luo, X. Examining the Callaway Model for Lattice Thermal Conductivity. *Phys. Rev. B: Condens. Matter Mater. Phys.* **2014**, *90*, 035203.

(5) Allen, P. B. Improved Callaway Model for Lattice Thermal Conductivity. *Phys. Rev. B: Condens. Matter Mater. Phys.* **2013**, *88*, 144302.

(6) Nath, P.; Plata, J. J.; Usanmaz, D.; Toher, C.; Fornari, M.; Buongiorno Nardelli, M.; Curtarolo, S. High Throughput Combinatorial Method for Fast and Robust Prediction of Lattice Thermal Conductivity. *Scr. Mater.* **2017**, *129*, 88−93.

(7) Tawfik, S. A.; Isayev, O.; Spencer, M. J. S.; Winkler, D. A. Predicting Thermal Properties of Crystals Using Machine Learning. *Adv. Theory Simul.* **2020**, *3*, 1900208.

(8) Juneja, R.; Singh, A. K. Guided Patchwork Kriging to Develop Highly Transferable Thermal Conductivity Prediction Models. *JPhys Mater.* **2020**, *3*, 024006.

(9) Zhu, T.; Gong, S.; Xie, T.; Gorai, P.; Grossman, J. C. Charting Lattice Thermal Conductivity of Inorganic Crystals. **2020**, arXiv:2006.11712. arXiv preprint.

(10) Wang, X.; Zeng, S.; Wang, Z.; Ni, J. Identification of Crystalline Materials with Ultra-Low Thermal Conductivity Based on Machine Learning Study. *J. Phys. Chem. C* **2020**, *124*, 8488−8495.

(11) Wan, X.; Feng, W.; Wang, Y.; Wang, H.; Zhang, X.; Deng, C.; Yang, N. Materials Discovery and Properties Prediction in Thermal Transport via Materials Informatics: A Mini Review. *Nano Lett.* **2019**, *19*, 3387−3395.

(12) Wei, H.; Bao, H.; Ruan, X. Genetic Algorithm-driven Discovery of Unexpected Thermal Conductivity Enhancement by Disorder. *Nano Energy* **2020**, *71*, 104619.

(13) Yan, J.; Wei, H.; Xie, H.; Gu, X.; Bao, H. Seeking for Low Thermal Conductivity Atomic Configurations in SiGe Alloys with Bayesian Optimization. *ES Energy Environ.* **2020**, *8*, 56−64.

(14) Yamada, H.; Liu, C.; Wu, S.; Koyama, Y.; Ju, S.; Shiomi, J.; Morikawa, J.; Yoshida, R. Predicting Materials Properties with Little Data Using Shotgun Transfer Learning. *ACS Cent. Sci.* **2019**, *5*, 1717−1730.

(15) Xiong, Z.; Cui, Y.; Liu, Z.; Zhao, Y.; Hu, M.; Hu, J. Evaluating Explorative Prediction Power of Machine Learning Algorithms for Materials Discovery Using K-fold Forward Cross-validation. *Comput. Mater. Sci.* **2020**, *171*, 109203.

(16) Carrete, J.; Li, W.; Mingo, N.; Wang, S.; Curtarolo, S. Finding Unprecedentedly Low-Thermal-Conductivity Half-Heusler Semiconductors via High-Throughput Materials Modeling. *Phys. Rev. X* **2014**, *4*, 011019.

(17) Carrete, J.; Li, W.; Mingo, N.; Wang, S.; Curtarolo, S. Finding Unprecedentedly Low-thermal-conductivity Half-Heusler Semiconductors via High-throughput Materials Modeling. *Phys. Rev. X* **2014**, *4*, 011019.

(18) Miller, S. A.; Gorai, P.; Ortiz, B. R.; Goyal, A.; Gao, D.; Barnett, S. A.; Mason, T. O.; Snyder, G. J.; Lv, Q.; Stevanović, V.; et al. Capturing Anharmonicity in a Lattice Thermal Conductivity Model for High-throughput Predictions. *Chem. Mater.* **2017**, *29*, 2494−2501.

(19) Plata, J. J.; Nath, P.; Usanmaz, D.; Carrete, J.; Toher, C.; de Jong, M.; Asta, M.; Fornari, M.; Nardelli, M. B.; Curtarolo, S. An Efficient and Accurate Framework for Calculating Lattice Thermal Conductivity of Solids: AFLOW-AAPL Automatic Anharmonic Phonon Library. *npj Comput. Mater.* **2017**, *3*, 45.

(20) Seko, A.; Togo, A.; Hayashi, H.; Tsuda, K.; Chaput, L.; Tanaka, I. Prediction of Low-thermal-conductivity Compounds with First-principles Anharmonic Lattice-dynamics Calculations and Bayesian Optimization. *Phys. Rev. Lett.* **2015**, *115*, 205901.

(21) Toher, C.; Plata, J. J.; Levy, O.; De Jong, M.; Asta, M.; Nardelli, M. B.; Curtarolo, S. High-throughput Computational Screening of Thermal Conductivity, Debye Temperature, and Grüneisen Parameter Using a Quasiharmonic Debye Model. *Phys. Rev. B: Condens. Matter Mater. Phys.* **2014**, *90*, 174107.

(22) van Roekeghem, A.; Carrete, J.; Oses, C.; Curtarolo, S.; Mingo, N. High-throughput Computation of Thermal Conductivity of High-temperature Solid Phases: the Case of Oxide and Fluoride Perovskites. *Phys. Rev. X* **2016**, *6*, 041061.

(23) Kresse, G.; Furthmüller, J. Efficient Iterative Schemes for Ab Initio Total-energy Calculations Using a Plane-wave Basis Set. *Phys. Rev. B: Condens. Matter Mater. Phys.* **1996**, *54*, 11169−11186.

(24) Kresse, G.; Joubert, D. From Ultrasoft Pseudopotentials to the Projector Augmented-wave Method. *Phys. Rev. B: Condens. Matter Mater. Phys.* **1999**, *59*, 1758−1775.

(25) Perdew, J. P.; Burke, K.; Ernzerhof, M. Generalized Gradient Approximation Made Simple. *Phys. Rev. Lett.* **1996**, *77*, 3865−3868.

(26) Monkhorst, H. J.; Pack, J. D. Special Points for Brillouin-zone Integrations. *Phys. Rev. B: Solid State* **1976**, *13*, 5188−5192.

(27) den Toonder, J. M. J.; van Dommelen, J. A. W.; Baaijens, F. P. T. The Relation Between Single Crystal Elasticity and the Effective Elastic Behaviour of Polycrystalline Materials: Theory, Measurement and Computation. *Modell. Simul. Mater. Sci. Eng.* **1999**, *7*, 909−928.

(28) Jia, T.; Chen, G.; Zhang, Y. Lattice Thermal Conductivity Evaluated Using Elastic Properties. *Phys. Rev. B* **2017**, *95*, 155206.

(29) Lu, L.; Anderson-Cook, C. M.; Robinson, T. J. Optimization of Designed Experiments Based on Multiple Criteria Utilizing a Pareto Frontier. *Technometrics* **2011**, *53*, 353−365.

(30) Banzhaf, W.; Nordin, P.; Keller, R. E.; Francone, F. D. *Genetic Programming*; Springer, 1998.

(31) Cage, P.; Kroo, I.; Braun, R. *Interplanetary Trajectory Optimization Using a Genetic Algorithm. Astrodynamics Conference*, 1994; p 3773.

(32) Fusting, C. Fast Symbolic Regression. https://github.com/cfusting/fast-symbolic-regression (accessed Oct 2019).

(33) Ouyang, R.; Curtarolo, S.; Ahmetcik, E.; Scheffler, M.; Ghiringhelli, L. M. SISSO: A Compressed-sensing Method for Identifying the Best Low-dimensional Descriptor in an Immensity of Offered Candidates. *Phys. Rev. Mater.* **2018**, *2*, 083802.

(34) Juneja, R.; Yumnam, G.; Satsangi, S.; Singh, A. K. Coupling High-throughput Property Map to Machine Learning for Predicting Lattice Thermal Conductivity. *Chem. Mater.* **2019**, *31*, 5145−5151.

(35) Gardner, M. W.; Dorling, S. R. Artificial Neural Networks (the Multilayer Perceptron)-a Review of Applications in the Atmospheric Sciences. *Atmos. Environ.* **1998**, *32*, 2627−2636.

(36) Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press, 2016.

(37) Hu, R.; Andreas, J.; Darrell, T.; Saenko, K. Explainable Neural Computation via Stack Neural Module Networks. *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018; pp 53−69.

(38) Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929−1958.

(39) Kingma, D. P.; Ba, J. Adam: A Method for Stochastic Optimization. **2014**, arXiv:1412.6980.

(40) Breiman, L. Bagging Predictors. *Mach. Learn.* **1996**, *24*, 123−140.

(41) Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825−2830.

(42) Wong, T.-T. Performance Evaluation of Classification Algorithms by *k*-fold and Leave-one-out Cross Validation. *Pattern Recognit.* **2015**, *48*, 2839−2846.

(43) Arlot, S.; Celisse, A. A survey of Cross-validation Procedures for Model Selection. *Stat. Surv.* **2010**, *4*, 40−79.

(44) Nembrini, S.; König, I. R.; Wright, M. N. The revival of the Gini Importance? *Bioinformatics* **2018**, *34*, 3711−3718.

(45) Fannjiang, C.; Listgarten, J. Autofocused Oracles for Model-based Design. **2020**, arXiv:2006.08052. arXiv preprint.

(46) Hernandez, A.; Balasubramanian, A.; Yuan, F.; Mason, S. A. M.; Mueller, T. Fast, Accurate, and Transferable Many-body Interatomic Potentials by Symbolic Regression. *npj Comput. Mater.* **2019**, *5*, 112.

(47) Schmidt, M.; Lipson, H. Distilling Free-Form Natural Laws from Experimental Data. *Science* **2009**, *324*, 81−85.

(48) Udrescu, S.-M.; Tegmark, M. AI Feynman: A Physics-inspired Method for Symbolic Regression. *Sci. Adv.* **2020**, *6*, No. eaay2631.

(49) Udrescu, S. M. AI-Feynman. https://github.com/SJ001/AI-Feynman (accessed June 2020).

(50) Udrescu, S. M.; Tan, A.; Feng, J.; Neto, O.; Wu, T.; Tegmark, M. AI Feynman 2.0: Pareto-optimal Symbolic Regression Exploiting Graph Modularity. **2020**, arXiv:2006.10782.

(51) Jacobs, R.; Mayeshiba, T.; Afflerbach, B.; Miles, L.; Williams, M.; Turner, M.; Finkel, R.; Morgan, D. The Materials Simulation Toolkit for Machine learning (MAST-ML): An Automated Open Source Toolkit to Accelerate Data-driven Materials Research. *Comput. Mater. Sci.* **2020**, *176*, 109544.

(52) Ward, L.; Dunn, A.; Faghaninia, A.; Zimmermann, N. E. R.; Bajaj, S.; Wang, Q.; Montoya, J.; Chen, J.; Bystrom, K.; Dylla, M.; et al. Matminer: An Open Source Toolkit for Materials Data Mining. *Comput. Mater. Sci.* **2018**, *152*, 60−69.