

PAPER • OPEN ACCESS

## A charge density prediction model for hydrocarbons using deep neural networks

To cite this article: Deepak Kamal *et al* 2020 *Mach. Learn.: Sci. Technol.* **1** 025003

View the [article online](#) for updates and enhancements.

### You may also like



- [Ab initio electronic structure calculations using a real-space Chebyshev-filtered subspace iteration method](#)  
Qiang Xu, Sheng Wang, Lantian Xue et al.
- [Plasmonic performance of Au, Ag, Cu, and Al alloys from many-body perturbation theory](#)  
Okan K Orhan and David D O'Regan
- [Difficulties in applying pure Kohn–Sham density functional theory electronic structure methods to protein molecules](#)  
Elias Rudberg



## PAPER

## A charge density prediction model for hydrocarbons using deep neural networks

## OPEN ACCESS

RECEIVED  
2 August 2019REVISED  
22 October 2019ACCEPTED FOR PUBLICATION  
19 November 2019PUBLISHED  
18 March 2020Deepak Kamal , Anand Chandrasekaran , Rohit Batra and Rampi Ramprasad

Department of Materials Science and Engineering, Georgia Institute of Technology, Atlanta, GA 30332, United States of America

E-mail: [rampi.ramprasad@mse.gatech.edu](mailto:rampi.ramprasad@mse.gatech.edu)**Keywords:** electron density, machine learning, hydrocarbons, density functional theory, neural networksSupplementary material for this article is available [online](#)

Original content from this work may be used under the terms of the [Creative Commons Attribution 3.0 licence](#).

Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.

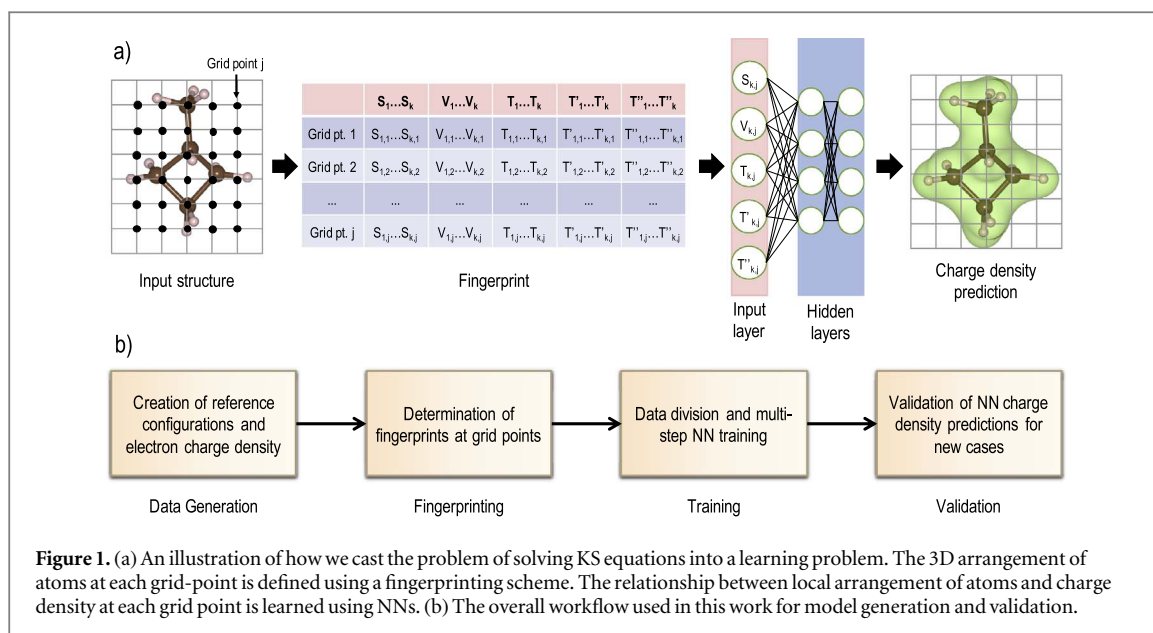
**Abstract**

The electronic charge density distribution  $\rho(r)$  of a given material is among the most fundamental quantities in quantum simulations from which many large scale properties and observables can be calculated. Conventionally,  $\rho(r)$  is obtained using Kohn–Sham density functional theory (KS-DFT) based methods. But, the high computational cost of KS-DFT renders it intractable for systems involving thousands/millions of atoms. Thus, recently there has been efforts to bypass expensive KS equations, and directly predict  $\rho(r)$  using machine learning (ML) based methods. Here, we build upon one such scheme to create a robust and reliable  $\rho(r)$  prediction model for a diverse set of hydrocarbons, involving huge chemical and morphological complexity (saturated, unsaturated molecules, cyclo-groups and amorphous and semi-crystalline polymers). We utilize a grid-based fingerprint to capture the atomic neighborhood around an arbitrary point in space, and map it to the reference  $\rho(r)$  obtained from standard DFT calculations at that point. Owing to the grid-based learning, dataset sizes exceed billions of points, which is trained using deep neural networks in conjunction with an incremental learning based approach. The accuracy and transferability of the ML approach is demonstrated on not only a diverse test set, but also on a completely unseen system of polystyrene under different strains. Finally, we note that the general approach adopted here could be easily extended to other material systems, and can be used for quick and accurate determination of  $\rho(r)$  for DFT charge density initialization, computing dipole or quadrupole, and other observables for which reliable density functional are known.

**1. Introduction**

The electronic charge density distribution  $\rho(r)$  of a molecule or a material is a physical observable of great significance. In principle, all ground state properties associated with a material can be accessed with the knowledge of its underlying  $\rho(r)$ , as per density functional theory (DFT) [1]. An accurate estimate of  $\rho(r)$  can provide insights concerning charge redistribution, bond formation etc, in molecular and materials systems.  $\rho(r)$  is also the starting point for a variety of electronic structure simulations aimed at calculating higher level electronic properties like electrostatic moments associated with molecules such as dipole and quadrupole moments, electrostatic potentials (recently demonstrated by Fabrizio *et al* [2]), electrostatic interaction energies etc. It can also be used to directly compute IR intensities [3] and identify binding sites in host-guest compounds [4–7].

Obtaining the charge density of a given material system is, however, non trivial. Although many-electron quantum chemistry methods can accurately estimate  $\rho(r)$ , they are extremely expensive. A modern less-expensive alternative is DFT, within the Kohn–Sham (KS) single-particle ansatz [8]. KS-DFT has become popular owing to its attractive cost-accuracy trade off, and serves as a vital part of the modern materials discovery portfolios [9–15]. But, KS-DFT is still expensive enough that high-performance computers are necessary to study large swaths of chemical spaces involving 10–100 thousands of molecules and materials. The primary



**Figure 1.** (a) An illustration of how we cast the problem of solving KS equations into a learning problem. The 3D arrangement of atoms at each grid-point is defined using a fingerprinting scheme. The relationship between local arrangement of atoms and charge density at each grid point is learned using NNs. (b) The overall workflow used in this work for model generation and validation.

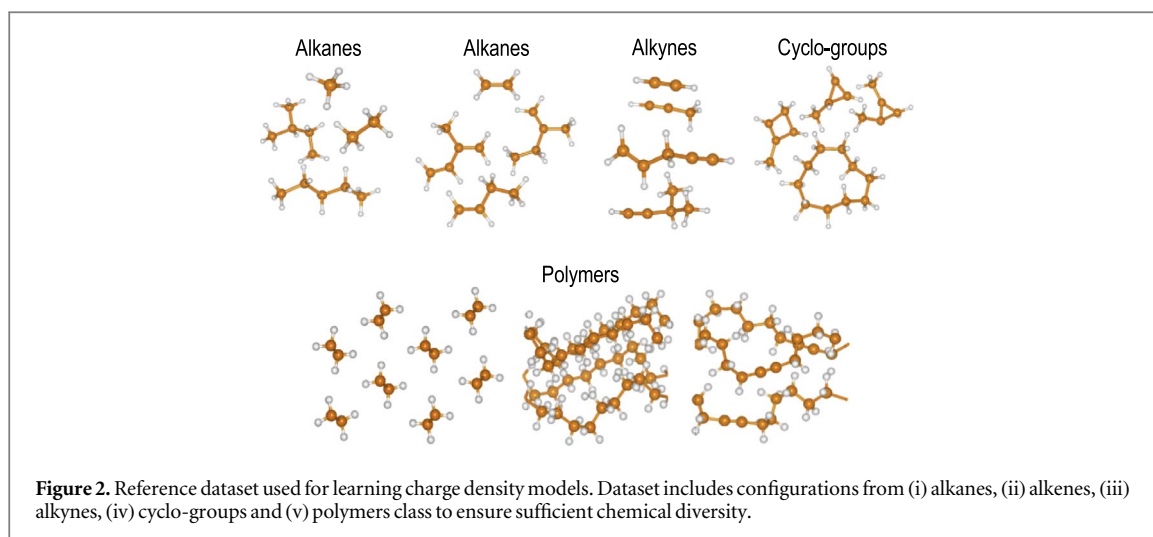
bottleneck of KS-DFT comes from the difficulty in self consistently solving the KS equation (specifically, having to orthogonalize single particle eigenvectors during the process), which produces the electronic charge density, eigenvalues and one-electron wave functions. It has recently been shown that this bottleneck may be sidestepped using machine learning (ML) techniques [16].

A majority of efforts related to application of ML in chemical and materials sciences have focused on learning a particular material property of interest (e.g. bandgap, formation energy, etc) using large reference databases of experimental measurements [17–19] or KS-DFT calculations [20–22]. A few of them go a step further and learn more fundamental properties like energies and atomic forces [23–33]. Lately, an increasing number of publications have tried to directly predict the ground state electronic charge density as well [28, 34–40]. This paper is a contribution towards the last class of efforts.

The Hohenberg–Kohn (H–K) theorem guarantees a unique mapping between charge density distribution and the structure of a material (in terms of nuclear potential), thereby suggesting a functional relationship between a structural representation and the corresponding charge density. Brockherde *et al* [37] showed that it is possible to map the total potential energy function of a given molecular structure to the associated charge density using a ML approach that mimics the H–K mapping. Starting with a representation of charge density in terms of plane wave basis functions, they successfully learned the relationship between the coefficients of these basis functions and the nuclear potentials using kernel ridge regression. Due to numerical representation of the structure in terms of full body nuclear potentials and use of a plane wave basis to represent the charge density, their model however offers limited transferability. Grisafi *et al* [39] overcome this challenge by expanding the total charge density as a sum of atom-centered non-orthogonal spherical harmonic basis functions and using symmetry-adapted Gaussian process regression to learn the coefficients of these basis functions. But the complex nature of the high dimensional regression problem that attempts to find atom-decomposed basis function coefficients limits the ability of the model to learn from a large dataset.

Here, we build on our previous work [41], which introduced a different approach to access the charge density, given the structure. A schematic of our approach is shown in figure 1(a). The idea is to map the charge density at every grid point to the respective local atomic environment around the grid point. In this work, we demonstrate the accuracy, transferability and scalability of this method, by constructing a general charge density model for hydrocarbons (involving nearly 60 chemically distinct molecules and polymers). A summary of the steps involved in the generation of this model is shown in figure 1(b). We start with generating a dataset which includes a large variety of hydrocarbon environments and the corresponding charge densities. Local atomic arrangement is then represented using a novel fingerprinting technique, introduced by Chandrasekaran *et al* [41]. The relationship between local environment and charge density is then learned using neural networks (NNs). To handle the problem of data explosion inherent to the nature of the approach, we introduce a step-by-step training process that exploits the ability of NNs to learn information and down-select huge datasets to computationally affordable subsets for efficient model training.

The charge density model thus obtained can readily be used to quickly estimate the electronic charge density of new configurations, and can even serve as a starting point for self-consistent computations involved in the KS-DFT routines, especially for systems of large sizes. Looking into the future, the model can also be used to



**Table 1.** Summary of the reference data set used in this work. The data set is categorized into different classes based on the types of bonding environments.

Group name	# of Molecules	# of Snapshots	# of grid points
Alkanes	10	100	204 472 320
Alkenes	20	200	568 604 800
Alkynes	11	110	322 884 480
Cyclo-groups	26	260	1061 442 560
Polymers	6	60	285 286 400

calculate properties like dipole and quadrupole moments of structures. Moreover, it can be utilized in conjunction with classical potentials or ML based force fields to access properties such as dielectric response.

## 2. Methodology

### 2.1. Dataset

In order to systematically capture a large variety of bonding environments prevalent in the hydrocarbon family, we have built a dataset comprising of five groups: (i) alkanes, (ii) alkenes, (iii) alkynes, (iv) cyclo-groups, and (v) polymers, as illustrated in figure 2. Such a classification ensures that a variety of possible bonding environments ( $sp^3$ ,  $sp^2$ ,  $sp^1$  and aromatic bonds) are well represented in the dataset. Groups (i), (ii) and (iii) contains all possible hydrocarbon molecules up to 5 carbon atoms. Additionally, the cyclo-group contains benzene and cyclo-alkanes up to 10 carbon atoms. The polymer class contains crystalline and amorphous polyethylene (PE), and PE with defects involving double bonds, triple bonds and side chains. Information about all the molecules/polymers present in each group is listed in appendix A. To generate non-equilibrium bonding environments, *ab initio* molecular dynamics (AI-MD) simulations were performed for each case, from which 10 snapshots were randomly sampled. Further, for each snapshot of each molecule, self-consistent field electronic structure optimization was performed to obtain the corresponding charge density to be utilized for the ML process. The distribution of the generated reference dataset in terms of number of molecules, snapshots per molecule, and the total number of configurations is presented in table 1.

### 2.2. DFT and MD details

AI-MD simulations for reference dataset generation, using a micro-canonical ensemble, were carried out at 300 K and for 0.5 ns with a time step of 1 fs. All charge density calculations in this work are done using the Vienna *Ab Initio* Simulation Package [42] (VASP) employing the Perdew–Burke–Ernzerhof exchange–correlation functional and the projector-augmented wave methodology. A Monkhorst–Pack grid with density of  $01 \text{ \AA}^{-1}$  adopted and a basis set of plane waves with kinetic energies up to 500 eV was used to represent the wave functions. Convergence studies were conducted to find the optimum K-points and other geometry-dependent numerical parameters for all calculations.

### 2.3. Fingerprint

Within a standard DFT implementation, the scalar field  $\rho(r)$  is calculated for a structure on a 3D grid of discrete points in space. Our objective is to find the relationship between  $\rho(r)$  at these grid points and the arrangement of atoms around them. We accomplish this using a *fingerprint* that numerically represents the atomic arrangement around a grid point [43]. Out of many powerful structure representations in literature [32, 38, 39, 43–45], we choose this *fingerprint* [43] because it systematically represents radial and angular distributions of the atoms around a point in space. This is crucial when using a grid based approach wherein large number of points are to be represented. Further, this representation is also more intuitive relative to the popular bispectrum approach [45]. The said fingerprint definition is based on our previous work, and consists of a hierarchy of scalar (S), vector (V) and tensor (T) components which capture the radial and angular distribution of atoms around a grid point. The scalar component that captures the radial information of atoms around a grid-point  $g$  using a predefined set of Gaussian functions ( $k$ ) of varying widths  $\sigma_k$  is defined as:

$$S_{k\Omega} = c_k \sum_{i=1}^{N_\Omega} \exp\left(\frac{-r_{gi}^2}{2\sigma_k^2}\right) f_c(r_{gi}), \quad (1)$$

where,  $r_{gi}$  is the distance between the atom  $i$  of specie  $\Omega$  and the reference grid-point  $g$ .  $N_\Omega$  denotes the number of atoms of type  $\Omega$ .  $f_c(r_{gi})$  is the cut-off function defined as  $0.5[\cos(\frac{\pi r_{gi}}{R_c}) + 1]$ , for  $r_{gi} \leq R_c$  and equal to 0 for  $r_{gi} > R_c$ . The coefficient  $c_k$  is the normalization constant given by  $(\frac{1}{\sqrt{2\pi}}\sigma_k)^3$ . The overall dimensionality of the scalar component is the product of Gaussian  $k$  and the number of element types. Similarly, the vector components are defined by

$$V_{k\Omega}^\alpha = c_k \sum_{i=1}^{N_\Omega} r_{gi}^\alpha \exp\left(\frac{-r_{gi}^2}{2\sigma_k^2}\right) f_c(r_{gi}), \quad (2)$$

$$T_{k\Omega}^{\alpha\beta} = c_k \sum_{i=1}^{N_\Omega} r_{gi}^\alpha r_{gi}^\beta \exp\left(\frac{-r_{gi}^2}{2\sigma_k^2}\right) f_c(r_{gi}), \quad (3)$$

where,  $\alpha$  and  $\beta$  represent the  $x$ ,  $y$  or  $z$  directions. Here pre-factors  $r_{gi}^\alpha$  and  $r_{gi}^\alpha r_{gi}^\beta$  are added to incorporate angular distribution of atoms of a specific species at a given radius  $r$ . Further to make vector and tensor components rotationally invariant, the following quantities are defined and used as the fingerprint components:

$$V_{k\Omega} = \sqrt{(V_{k\Omega}^x)^2 + (V_{k\Omega}^y)^2 + (V_{k\Omega}^z)^2}, \quad (4)$$

$$T_{k\Omega} = T_{k\Omega}^{xx} + T_{k\Omega}^{yy} + T_{k\Omega}^{zz}, \quad (5)$$

$$T_{k\Omega}' = T_{k\Omega}^{xx} T_{k\Omega}^{yy} + T_{k\Omega}^{yy} T_{k\Omega}^{zz} + T_{k\Omega}^{zz} T_{k\Omega}^{xx} - (T_{k\Omega}^{xy})^2 - (T_{k\Omega}^{yz})^2 - (T_{k\Omega}^{zx})^2, \quad (6)$$

$$T_{k\Omega}'' = \text{determinant}(T_{k\Omega}^{\alpha\beta}). \quad (7)$$

Collecting the different pieces, the overall fingerprint consists of terms  $S_{k\Omega}$ ,  $V_{k\Omega}$ ,  $T_{k\Omega}$ ,  $T_{k\Omega}'$  and  $T_{k\Omega}''$ . By construction, each of the fingerprint component is invariant to system translation, rotation and permutations of atoms, similar to the target property  $\rho(r)$ . To calculate these fingerprints at a grid point, we use a set of predefined Gaussian functions with its mean at the chosen grid point and having a range of standard deviations ( $\sigma_k$ ). A total of 16 Gaussian functions were used in this study with widths varying from 0.25 to 8 Å divided on a logarithmic scale, along with a cut-off parameter of  $R_c = 9$  Å.

### 2.4. ML method

The choice of ML algorithm is another key ingredient in developing any ML based model. Since the problem at hand demands handling large volumes of data, amounting to billions of points, conventional methods like Gaussian process regression are computationally inefficient. Even varieties of sparse Gaussian process regression based methods [46–48] effectively reduce the train and prediction cost to  $\mathcal{O}(m^2 \times n)$  or  $\mathcal{O}(m^2)$  [48], where  $m$  is the size of *psuedo inputs* and  $n$  is the number of data points. NNs have been the ‘go to’ solution for big data problems in situations where the functional relationships are completely unknown and when the dataset size is enormous. They have been widely used in areas like image-recognition, targeted advertising and natural language processing for at least a decade [49]. Moreover, within the materials science and chemistry communities, NNs have been used extensively to develop force-fields for a variety of molecular and materials systems [25, 26, 32, 38].

Here, we employ a type of NN called deep feed forward neural network (DNN). The architecture is composed of a set of layers of neurons, namely, the input-layer, hidden layers and an output layer. In the input layer, each neuron contains a component of the fingerprint vector (see figure 1(a)) and the size of the input-layer is decided by the number of components in the fingerprint. In the hidden layers, each neuron represents a

predefined parametrized function acted upon by a weighted linear transform of each of the neurons in the previous layer. The number of neurons in every hidden layer and number of hidden layers themselves are hyperparameters chosen/optimized based on the problem at hand. Here, for convenience, we keep the number of neurons in each layer as a constant (128 neurons per layer) and vary the number of hidden layers to study its dependence on model performance. The output layer has a single neuron which collects the output from the last hidden layer and calculates the property of interest ( $\rho(r)$ , in this work) using a weighted linear transform. Using this method, the underlying functional relationship between the fingerprint and the property is learned by finding the correct weights of the linear transforms and the function parameters used in each neuron by an optimization algorithm [50].

In this work, Keras [51] with Tensorflow backend was used to build the models. The input layer and hidden layers have 128 neuron each while the output layer has one neuron. Relu activation function is used in the hidden layers. A mini-batch training algorithm with random sampling was employed in training all models, root mean square error (RMSE) was used as the objective function, and the Adam optimizer was used to optimize the weights. NN hyperparameters like the number of nodes per layer, the activation function used in the neurons, the decision of optimizers used, the form of the loss function used, and the stopping criterion for training hyperparameters were chosen based on validation error minimization criterion and time taken to train the NN model [41].

## 2.5. Model training

In our grid based approach, the charge density at every grid-point was used as input to train our model. To maintain the accuracy of the model, a fine grid-spacing of approximately 0.1 Å was used. This results in an enormous amount of training data of the order of billions of examples (for the whole dataset), which raises many practical challenges in handling the learning process. To reduce the size of the problem, a multi-step process was employed in developing of the model. The dataset was first divided into subsets as described in the following section. The models were then progressively improved by training on different subsets in a sequential manner as described below, closely following the idea of incremental learning. We note here that many of the advanced mini-batch diversification techniques [52, 53] which focus on intelligently dividing the datasets in smaller subsets to allow NN training at low computational (memory) costs, could also be a solution. They do not down-select or throw away points from datasets, but utilize all points for NN training. We chose to go with down-selecting the data instead owing to the similarity between examples in the dataset and other practical considerations like time taken to train the NN.

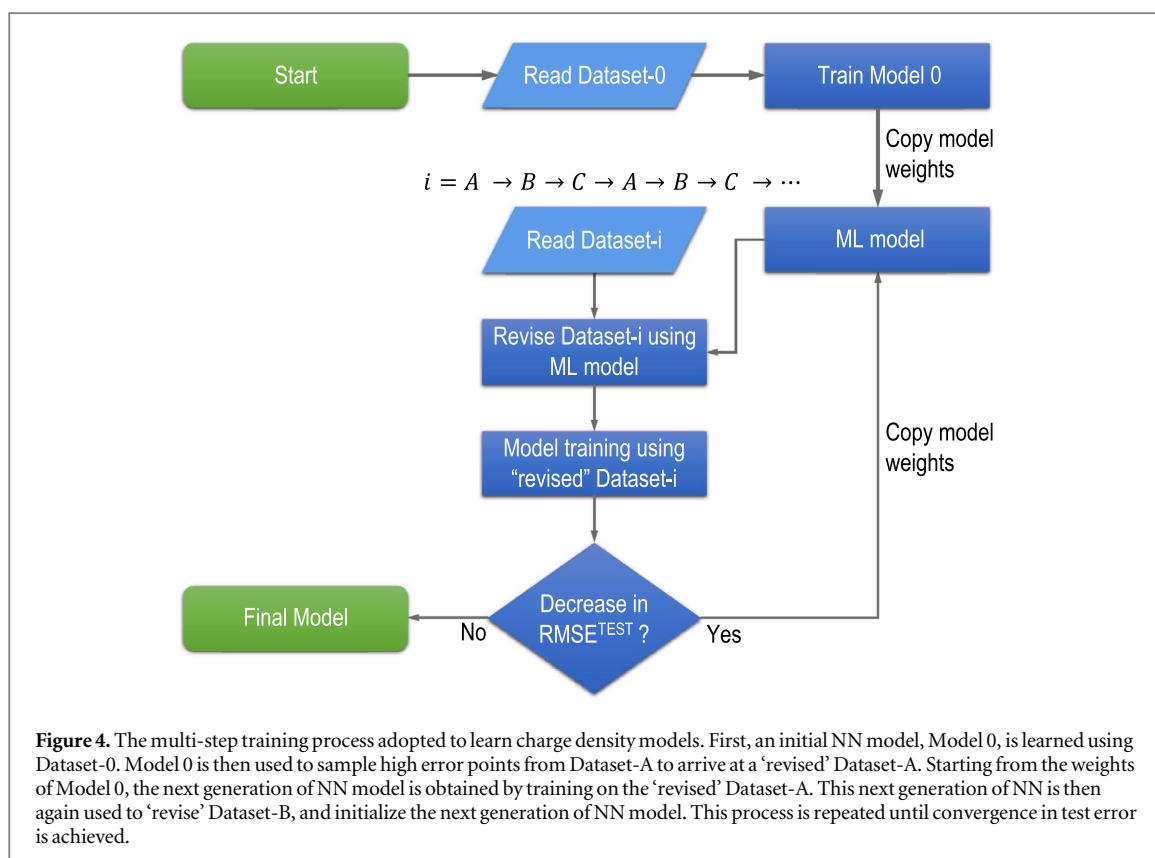
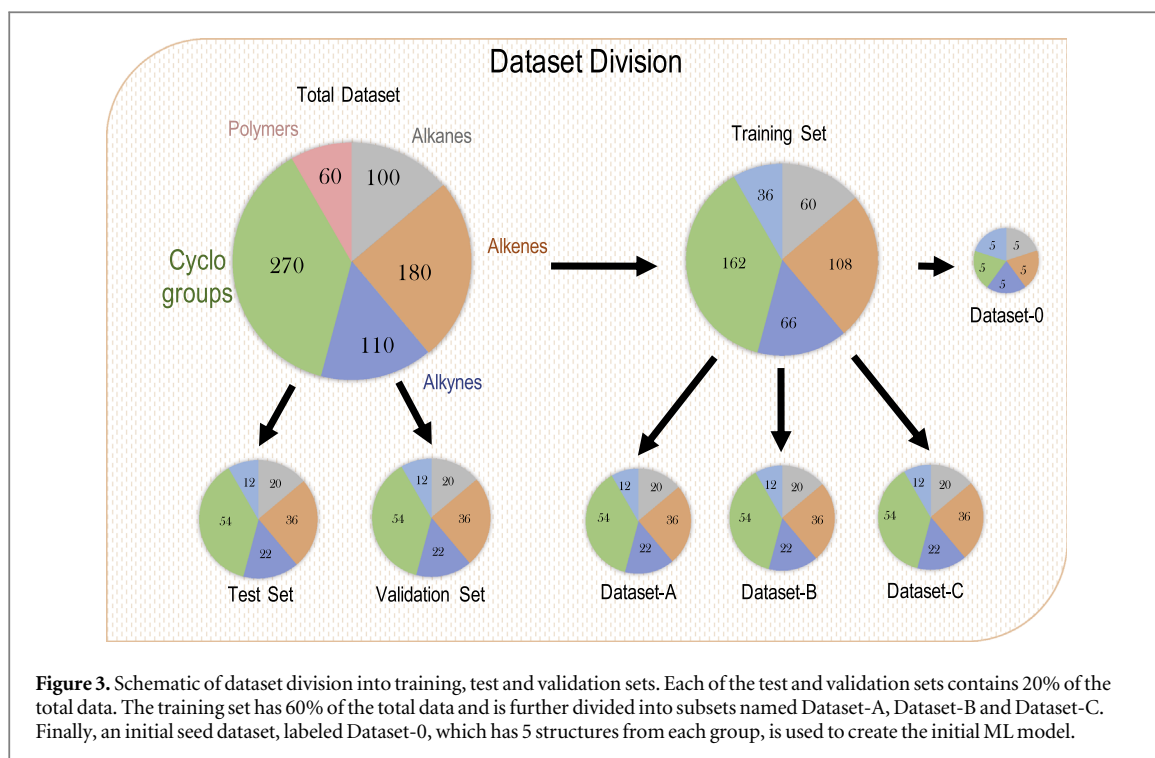
### 2.5.1. Dataset division

To start with, the dataset is divided into: the training set, the validation set and the test set as shown in figure 3. The examples used to train the NNs were drawn from the training set, whereas the validation set was used to prevent over-fitting and the test set was used to evaluate the model performance. To overcome practical limitations in handling data, the training dataset is further divided into 3 equal subsets, i.e. Dataset-A, Dataset-B and Dataset-C. In addition to these, a much smaller dataset, Dataset-0, is introduced to initiate the training workflow. For all data subsets, an equal proportion of data is taken from each of the five hydrocarbon groups (see figure 4) to ensure enough chemical diversity, and to facilitate continual improvement of the model during training.

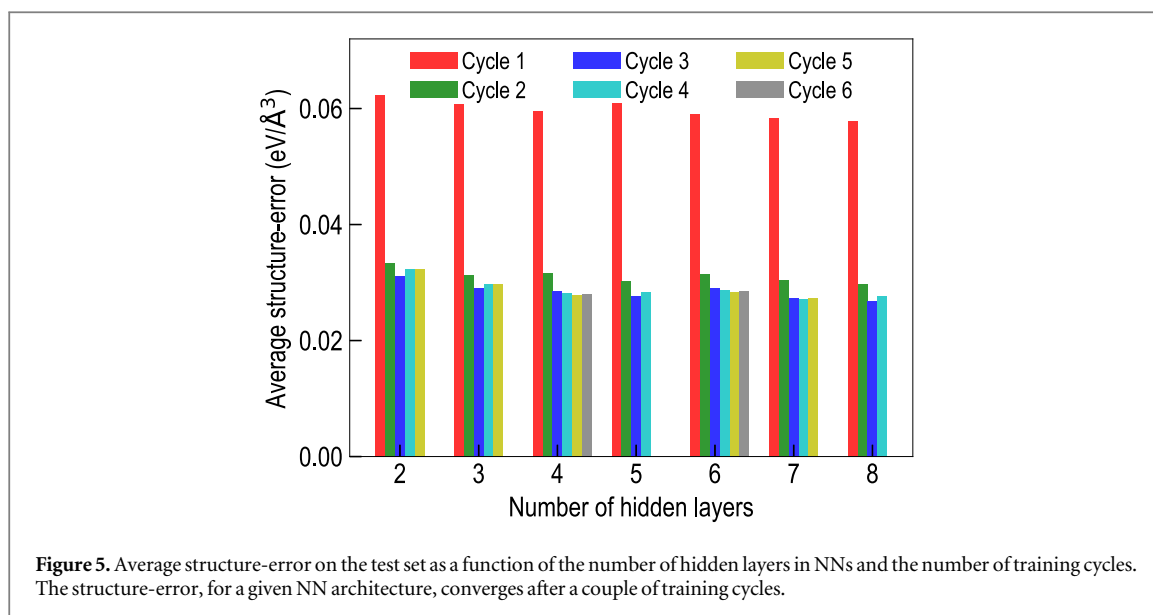
### 2.5.2. Multi-step model training

Since the hardware limits the number of examples that can be learned during the training, we expose our model to different data subsets (and thus to the entire dataset) in a sequential manner. Further, we use the pre-trained models to find data points where our model performs poorly, and then specifically sample from such high error points to train in a loop-wise fashion. This is one of the possible active sampling methods, very similar in spirit to adaboost learning [54]. As demonstrated in the figure 4, we start by creating an initial NN model, termed Model 0, using the relatively smaller Dataset-0 (which amounts to about 20 million examples). The Model 0 is next used to make predictions on Dataset-A, and exclusively find data points for which the model performance is poor. These high error points are then used to form the 'revised' Dataset-A, details of which are provided in appendix B. This idea of retaining only the data points with high errors allows us to limit the size of training examples. The next generation of NN model (with the same NN architecture) initialized using weights of Model 0 is then trained using the 'revised' Dataset-A. The obtained NN is then again used to 'revise' Dataset-B and initialize weights of the future NN model, with the process repeated until convergence in test errors is obtained. Further, in order to expose our NN model to the entire dataset, the data subsets are revised in a cyclic manner with the order  $A \rightarrow B \rightarrow C \rightarrow A \rightarrow B \rightarrow C$ , as reflected in figure 4. RMSE between the DFT calculated and model predicted values was chosen as the error metric.





This multi-step workflow has several key advantages. Initial sampling and subsequent revision of data subsets reduces the practical difficulties in handling data by reducing the dataset size. This also prevents the models from being biased due to the presence of large number of examples from a small region, which usually results in poor representation of outliers in the model. The initialization of weights from the previous generation of NN models, helps to efficiently transfer knowledge learned from the past data, a process termed as transfer learning. This allows to further improve the model performance for the new examples, without having to access the past reference data.



### 3. Results and discussion

#### 3.1. Effect of model architecture on performance

Past studies [55, 56] indicate a heavy dependence of the model performance on the depth of the NN model. Here, to understand the effect of number of hidden layers on the performance of the model and to choose the best NN model that can be trained using the prescribed methodology, a set of different NN architectures were explored. Models with minimum cross-validation RMSE error were chosen to prevent over-fitting. RMSE of the test set was used to evaluate these models. To limit the search space of hyper parameters, the number of neurons on all hidden layers were kept fixed as described earlier. Figure 5 shows the performance of models made with varying numbers of hidden layers when tested on the test set. To compare the performance of the models more clearly, we report an average test set error computed by taking the mean of the error of 1000 points with maximum absolute error for each structure (reported as structure-error in figure 5). The model developed using a single hidden layer was found to show significantly high errors and hence is not reported here. A decrease in structure-error with increasing number of hidden layers is evident from the figure.

Figure 5 indicates that all models are able to systematically learn the information contained in different datasets following the proposed multi-step methodology. It should be noted that models developed in each cycle with a particular architecture do not have access to the entire dataset, but only to one of the Dataset-A, B and C. Information about the previously seen data is only contained in the initial weights from which the model training begins.

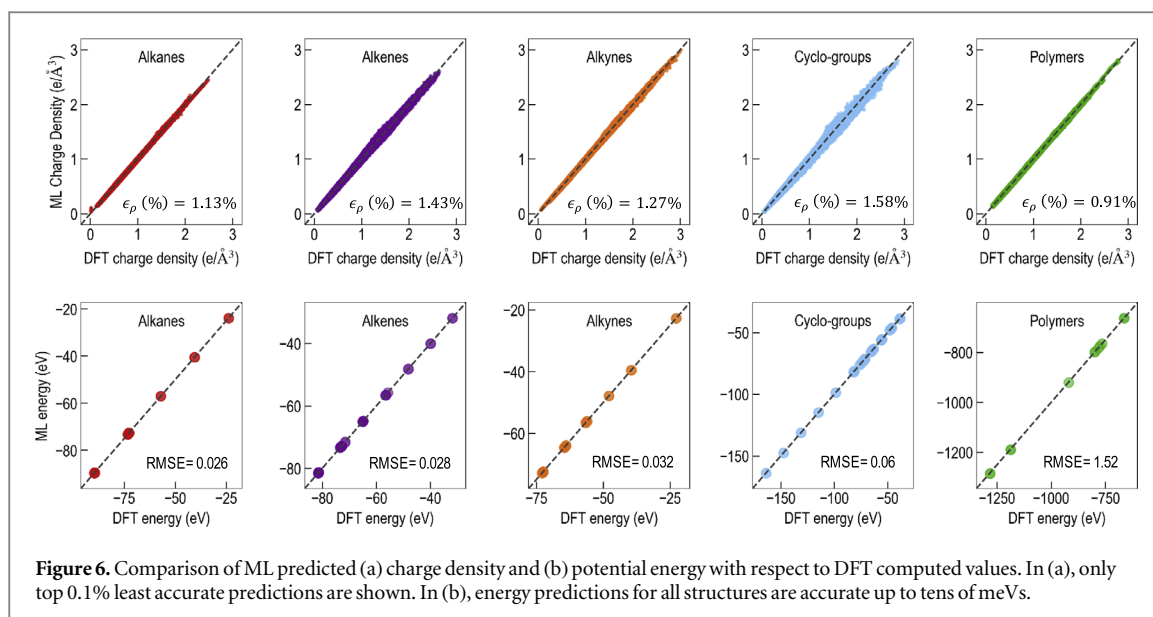
The proposed methodology thus provides a way to systematically train models on very large datasets by splitting them into small batches and choosing training points which are unfamiliar to the model. This methodology also enables us to improve a pre-trained model by incorporating new environments of interest without having access to the data used to train the model. On comparing the performance of the best models as a function of increasing depth of the NN, it can be seen that there is no significant effect on the performance with the increase in model depth. Rather, the performance is more dependant on the number of training cycles. This means that all these NN models are able to capture the underlying functional relationship between our fingerprint and the charge density fairly well.

#### 3.2. Performance of best model

To evaluate the performance of the model on different classes of structures in the test dataset, the best model (in terms of charge density RMSE) out of all those examined earlier was chosen (hidden layers:8, cycle:2). Figure 6 shows parity plots between the charge density predicted using ML and those calculated using DFT. Owing to the large size of the dataset, only charge densities of the top 1% least accurate points from each structure in the test set is depicted in the figure 6(a).

In order to estimate the accuracy of the prediction and compare it with other works in literature, % weighed mean absolute percentage error ( $\epsilon_{\rho}(\%)$ ), was computed.





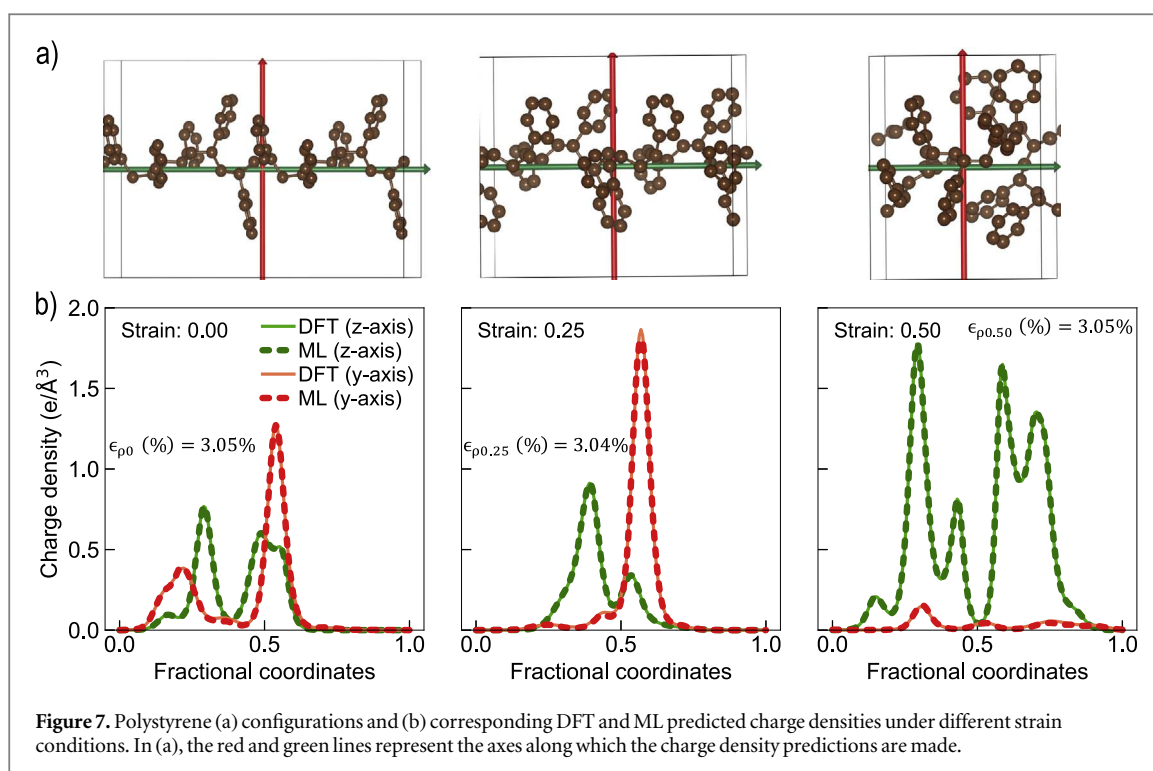
$$\epsilon_{\rho}(\%) = 100 \times \frac{1}{N_e} \sum_i^n N_e^i \frac{\int dr |\rho_{\text{DFT}}^i(r) - \rho_{\text{ML}}^i(r)|}{\int dr \rho_{\text{DFT}}^i(r)}, \quad (8)$$

where  $\rho_{\text{DFT}}^i(r)$  is the charge density of test structure  $i$  calculated using DFT,  $\rho_{\text{ML}}^i(r)$  is the charge density of test structure  $i$  predicted by the ML model, and  $N_e^i$  is the number of electrons in each structure and  $N_e$  is the total number of electron.  $\epsilon_{\rho}(\%)$  was found to be 1.26% for the entire test set. This is very close to the errors reported in other studies [2, 39]. The accuracy of the prediction can be visually observed from figure 6 where that charge density in each class of structures contained in the test set are plotted.  $\epsilon_{\rho}(\%)$  for each subclass of structures is also reported in the figure. The skewness in the parity plots, seen pronounced for alkenes and cyclo-groups suggest that errors are lower when charge density is low, that is, close to the atom centers and far away from atom centers. This is intuitive as for these grid points, the charge density is not as heavily dependant on the arrangement of atoms as compared to regions between atoms. For regions between the low valence density points, the errors are higher, but still close to the parity line.

To assess if charge density with this level of accuracy is meaningful, we evaluated energy of the structures in each group using the ML-predicted charge density and compared it with those obtained using self consistent field converged calculations (SCF) with VASP. We did this by predicting charge density on the same fast Fourier transformation grid obtained from these SCF calculations using our model and the values in a format which can be read in by VASP. Further, we used VASP to evaluate the energy, keeping the charge density constant. Parity plots for this is shown in figure 6(b). For all chosen classes of molecules and polymers, the correlation between the DFT calculated and ML predicted energy values are close to unity. These plots provide additional reassurance of the agreement between DFT and predicted charge densities for all chosen classes of hydrocarbons. It was also found that when the  $\rho(r)$  calculated by the final model is used as initial guess in VASP, approximately 10% speedup was observed for SCF convergence, for a small structure (<100 atoms). This % speedup is expected to increase as a function of system size.

### 3.3. Tests on unseen structures

To test the transferability of the developed model to similar environments, charge density predictions were done on a set of totally unseen structures. A single chain of polystyrene was created and compressively strained to different degrees along the chain axis, for this test. Note that the training set contains PE and benzene as two different entities but does not have the structure of polystyrene itself. Figure 7 shows an illustration of three of these structures with 0%, 25% and 50% strains. Charge densities were calculated for these structures using one self consistent loop DFT calculation. These charge densities were then compared to those predicted by the ML model. Figure 7(b) shows line plots of charge density indicated along the axis described in figure 7(a). A good agreement between the model predictions and DFT computations is can be observed visually and looking at the percentage absolute error for each structure ( $\epsilon_{\rho,0}(\%)$ ,  $\epsilon_{\rho,0.25}(\%)$ ,  $\epsilon_{\rho,0.50}(\%)$ ) in figure 7. For an even closer inspection, a difference plot between the predicted and the calculated charge densities is reported in appendix C (figure C1). However, the energy values computed using the charge density predicted by the ML model differed by a few eVs, signaling that the model should be retrained using some instances of polystyrene to reliably estimate energy values using the model.



### 3.4. Conclusions

In summary, we have successfully demonstrated a ML based approach to build surrogate charge density models (that circumvent a direct quantum mechanical calculation) using reference datasets, obtained from first principles DFT calculations. The robustness and transferability of the scheme was established using the example of a fairly diverse (chemically) class of material, i.e. hydrocarbons. A grid-based fingerprint that captures the 3D atomic neighborhood distribution of local environments was used to learn respective grid-based charge density value, which resulted in a need to handle huge amounts of data (greater than billion points). Such a large dataset was trained using a multi-step iterative scheme that systematically learns from small chunks of high error points, similar in spirit to adaboost learning [57]. The scheme was shown to produce accurate charge density predictions for new or unseen hydrocarbon systems, and recovered correct energies when SCF DFT calculations were initiated from the model predicted charge densities, thereby providing a pathway to accelerate quantum mechanical simulations through better initialization of charge densities (instead of conventional charge density initializations [58]). Further, we believe that the present scheme could be used to compute other charge density related properties, such as dipole, quadrupole moments, etc and other properties for which density functionals may be available. Finally, we note that the approach introduced here is general and can be effortlessly repeated for other material systems to make accurate charge density models.

### Funding

This work is supported by the Office of Naval Research through N0014-17-1-2656, a Multi-University Research Initiative (MURI) grant. The authors thank XSEDE for the utilization of Stampede2 cluster via project ID 'DMR080058N'.

### Competing interests

The authors declare that they have no competing financial interests.

### Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request. All data used to generate the models (and the train-validation-split details) are available online at available online at <https://khazana.gatech.edu>.

## Appendix A. Dataset

This section lists all distinct chemical species (molecules/polymers) contained in each group in the dataset.

### A.1. Alkanes

methane, ethane, propane, butane (2 isomers), 2-methyl propane, but-1-ene, pentane, 2-methyl-butane, 2-2-dimethyl-propane.

### A.2. Alkenes

ethene, propene, propadiene, but-1-ene, but-2-ene, 2-methyl, propene, but-1-2-diene, but-1-3-diene, pent-1-ene, pent-2-ene, 2-methyl-but-1-ene, 2-methyl-but-2-ene, 3-methyl-but-1-ene, penta-1-2-diene, penta-1-4-diene, penta-2-3-diene, penta-1-3-diene, 2-methyl-but-1-3-diene, 3-methyl-but-1-2-diene.

### A.3. Alkynes

ethyne, propyne, but-1-yne, but-2-yne, but-3-en-1-yne, pent-1-yne, pent-2-yne, 3-methyl-but-1-yne, penta-3-en-1-yne, penta-4-en-1-yne, penta-4-en-2-yne

### A.4. Cyclo-groups

cyclopropane, cyclopropene, 1-methyl-cyclopropene, 3-meth-3-en-cyclopropene, 3-methyl-cyclopropene, 1-methyl-cyclopropane, cyclobutene, cyclobutadiene, 1-1-dimethyl-cyclopropane, methyl-cyclobutane, cyclopentene, methyl-cyclobutadiene, 1-2-dimethyl-cyclopropane, 1-2-dimethyl-cyclopropene, 1-3-dimethyl-cyclopropene, 1-methyl-cyclobutene, 3-3-dimethyl-propene, 3-meth-en-cyclobut-1-ene, 3-methyl-cyclobutene, cyclopenta-1-2-diene, cyclohexane, benzene, cycloheptane, cyclooctane, cyclononane, cyclodecane.

### A.5. Polymers

polyethylene crystalline, polyethylene disordered, polyethylene with double bond defects, polyethylene with triple bond defects, polyethylene with -CH<sub>3</sub> side chain defects.

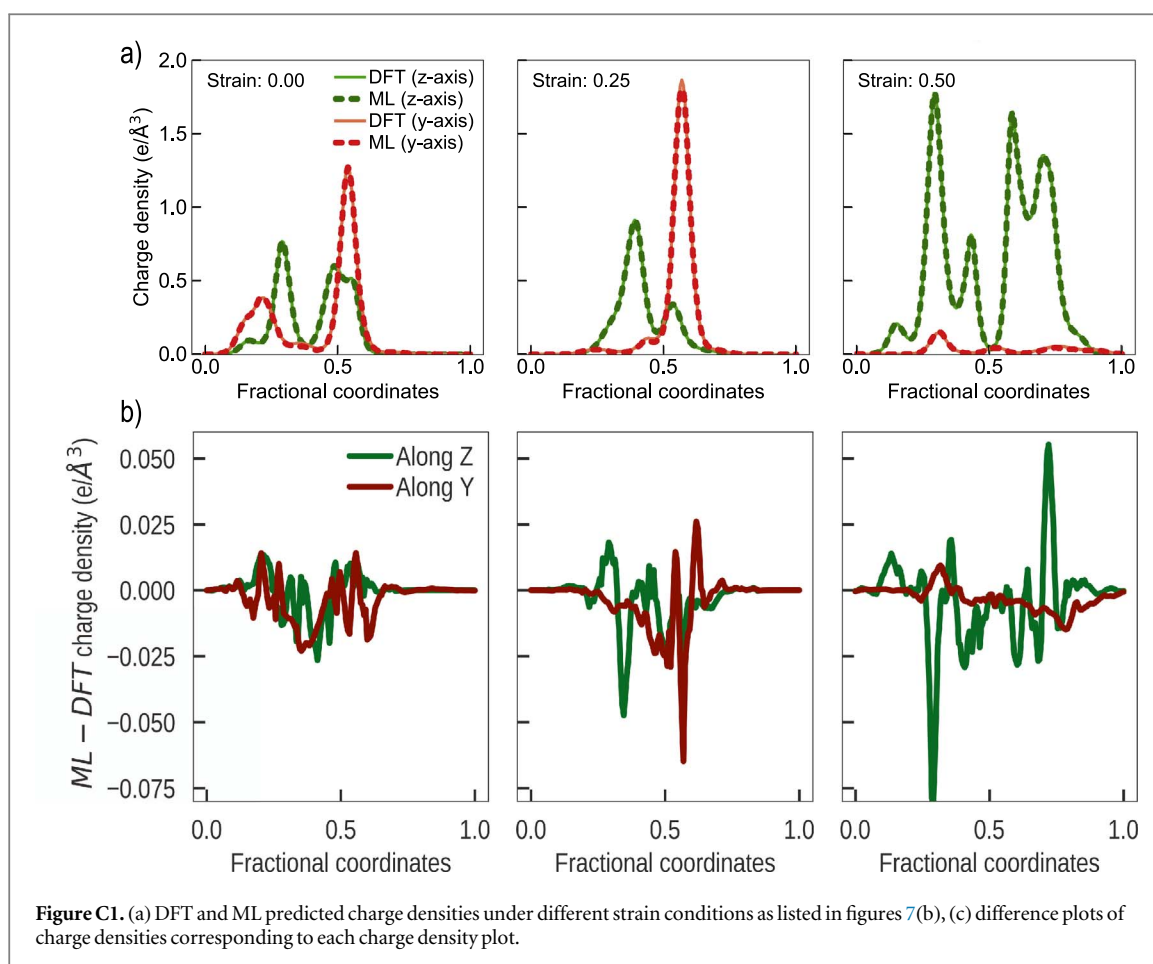
## Appendix B. Data revision

Data revision was introduced to address the following:

- (i) to purge practical difficulties in handling a huge amount of data
- (ii) to eliminate bias in the model due to domination of examples from most prolific training points
- (iii) as a method to improve the model given new examples without having to access the past reference data

Dataset revision refers to the process of choosing a small subset from a given large dataset. Examples to be included in this subset are chosen based on the performance of the model from the previous iteration (or initial model), as described in figure 4 in the main text, on examples of the large dataset. For instance, Model: (i-1), a model trained in a previous training cycle is used to predict the charge densities at each grid point in each structure in training dataset-i. Then, the absolute difference between the predicted values and reference values is evaluated for each example. These errors can be loosely categorized as a gaussian function with an abnormally sharp peak centered around 0. Now, all points whose error values fall half a standard deviation (of the fitted gaussian function) away from 0 are grouped as High Error points. 75% of points from this high error dataset are then replaced with points from the low error regions to form the Revised Dataset. 75% are chosen from the well predicted region because it was found that the performance of the model worsens if it is trained on high percentage examples of high error regions. These revised dataset are used for training the model within each training cycle.

## Appendix C. Charge density difference plot for polystyrene



## ORCID iDs

Deepak Kamal  <https://orcid.org/0000-0003-1943-7774>

Anand Chandrasekaran  <https://orcid.org/0000-0002-2794-3717>

## References

- [1] Hohenberg P and Kohn W 1964 *Phys. Rev.* **136** B864–71
- [2] Fabrizio A, Grisafi A, Meyer B, Ceriotti M and Corminboeuf C 2019 *Chem. Sci.* **10** 9424–32
- [3] Porezag D and Pederson M R 1996 *Phys. Rev. B* **54** 7830
- [4] Buckingham A, Fowler P and Hutson J M 1988 *Chem. Rev.* **88** 963–88
- [5] Castleman A Jr and Hobza P 1994 *Chem. Rev.* **94** 1721–2
- [6] Brutschy B and Hobza P 2000 *Chem. Rev.* **100** 3861–2
- [7] Hobza P and Rezac J 2016 *Chem. Rev.* **116** 4911–2
- [8] Kohn W and Sham L J 1965 *Phys. Rev.* **140** A1133
- [9] Becke A D 2014 *J. Chem. Phys.* **140** 18A301
- [10] Jones R O 2015 *Rev. Mod. Phys.* **87** 897
- [11] Burke K 2012 *J. Chem. Phys.* **136** 150901
- [12] Jain A, Shin Y and Persson K A 2016 *Nat. Rev. Mater.* **1** 15004
- [13] Mannodi-Kanakkithodi A, Treich G M, Huan T D, Ma R, Tefferi M, Cao Y, Sotzing G A and Ramprasad R 2016 *Adv. Mater.* **28** 6277–91
- [14] Batra R, Huan T D, Jones J L, Rossetti G Jr and Ramprasad R 2017 *J. Phys. Chem. C* **121** 4139–45
- [15] Chen L, Huan T D and Ramprasad R 2017 *Sci. Rep.* **7** 6128
- [16] Brockherde F, Vogt L, Li L, Tuckerman M E, Burke K and Müller K R 2017 *Nat. Commun.* **8** 872
- [17] Kim C, Chandrasekaran A, Jha A and Ramprasad R 2019 *MRS Commun.* **9** 860–6
- [18] Mannodi-Kanakkithodi A, Chandrasekaran A, Kim C, Huan T D, Paliana G, Botu V and Ramprasad R 2018 *Mater. Today* **21** 785–96
- [19] Kim C, Chandrasekaran A, Huan T D, Das D and Ramprasad R 2018 *J. Phys. Chem. C* **122** 17575–85
- [20] Mannodi-Kanakkithodi A, Chandrasekaran A, Kim C, Huan T D, Paliana G, Botu V and Ramprasad R 2018 *Mater. Today* **21** 785–96
- [21] Balachandran P V, Emery A A, Gubernatis J E, Lookman T, Wolverton C and Zunger A 2018 *Phys. Rev. Mater.* **2** 043802
- [22] Kim C, Chandrasekaran A, Doan Huan T, Das D and Ramprasad R 2018 *J. Phys. Chem. C* **122** 17575–85
- [23] Botu V and Ramprasad R 2015 *Int. J. Quantum Chem.* **115** 1074–83
- [24] Botu V and Ramprasad R 2015 *Phys. Rev. B* **92** 094306
- [25] Behler J and Parrinello M 2007 *Phys. Rev. Lett.* **98** 146401
- [26] Behler J 2011 *J. Chem. Phys.* **134** 074106

- [27] Bartók A P, Payne M C, Kondor R and Csányi G 2010 *Phys. Rev. Lett.* **104** 136403
- [28] Schütt K, Kindermans P J, Felix H E S, Chmiela S, Tkatchenko A and Müller K R 2017 SchNet: a continuous-filter convolutional neural network for modeling quantum interactions *Advances in Neural Information Processing Systems* **30** 992–1002
- [29] Botu V, Batra R, Chman J and Ramprasad R 2016 *J. Phys. Chem. C* **121** 511–22
- [30] Kolb B, Lentz L C and Kolpak A M 2017 *Sci. Rep.* **7** 1192
- [31] Huan T D, Batra R, Chman J, Krishnan S, Chen L and Ramprasad R 2017 *NPJ Comput. Mater.* **3** 37
- [32] Smith J S, Isayev O and Roitberg A E 2017 *Chem. Sci.* **8** 3192–203
- [33] Imbalzano G, Anelli A, Giofré D, Klees S, Behler J and Ceriotti M 2018 *J. Chem. Phys.* **148** 241730
- [34] Snyder J C, Rupp M, Hansen K, Müller K R and Burke K 2012 *Phys. Rev. Lett.* **108** 253002
- [35] Montavon G, Rupp M, Gobre V, Vazquez-Mayagoitia A, Hansen K, Tkatchenko A, Müller K R and Von Lilienfeld O A 2013 *New J. Phys.* **15** 095003
- [36] Schütt K T, Glawe H, Brockherde F, Sanna A, Müller K R and Gross E K U 2014 *Phys. Rev. B* **89** 205118
- [37] Brockherde F, Vogt L, Li L, Tuckerman M E, Burke K and Müller K R 2017 *Nat. Commun.* **8** 872
- [38] Schütt K T, Sauceda H E, Kindermans P J, Tkatchenko A and Müller K R 2018 *J. Chem. Phys.* **148** 241722
- [39] Grisafi A, Fabrizio A, Meyer B, Wilkins D M, Corminboeuf C and Ceriotti M 2018 *ACS Cent. Sci.* **5** 57–64
- [40] Fowler A T, Pickard C J and Elliott J A 2019 *J. Phys.: Mater.* **2** 034001
- [41] Chandrasekaran A, Kamal D, Batra R, Kim C, Chen L and Ramprasad R 2019 *NPJ Comput. Mater.* **5** 22
- [42] Kresse G and Furthmüller J 1996 *Phys. Rev. B* **54** 11169
- [43] Batra R, Tran H D, Kim C, Chapman J, Chen L, Chandrasekaran A and Ramprasad R 2019 *J. Phys. Chem. C* **123** 15859–66
- [44] Willatt M J, Musil F and Ceriotti M 2019 *J. Chem. Phys.* **150** 154110
- [45] Behler J 2016 *J. Chem. Phys.* **145** 170901
- [46] Quiñonero-Candela J and Rasmussen C E 2005 *J. Mach. Learn. Res.* **6** 1939–59
- [47] Quiñonero-Candela J et al 2010 *J. Mach. Learn. Res.* **11** 1865–81
- [48] Snelson E and Ghahramani Z 2006 Sparse Gaussian processes using pseudo-inputs *NIPS'05: Proc. of the 18th Int. Conf. on Neural Information Processing Systems* 1257–64
- [49] LeCun Y, Bengio Y and Hinton G 2015 *Nature* **521** 436
- [50] Hecht-Nielsen R 1992 Theory of the backpropagation neural network *Neural Networks for Perception* (Amsterdam: Elsevier) pp 65–93
- [51] Chollet F et al 2015 Keras (<https://keras.io>)
- [52] Zhang C, Öztireli C, Mandt S and Salvi G 2019 Active mini-batch sampling using repulsive point processes *Proc. AAAI Conf. on Artificial Intelligence* vol 33, pp 5741–8
- [53] Zhang C, Kjellstrom H and Mandt S 2017 arXiv:1705.00607
- [54] Hastie T, Rosset S, Zhu J and Zou H 2009 *Stat. Interface* **2** 349–60
- [55] He K, Zhang X, Ren S and Sun J 2016 Deep residual learning for image recognition *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* pp 770–8
- [56] Eldan R and Shamir O 2016 The power of depth for feedforward neural networks *Conf. on Learning Theory* pp 907–40
- [57] Schire R E and Freund Y 2013 *Kybernetes* **42** 164–6
- [58] Woods N, Hasnip P and Payne M 2019 *J. Phys.: Condens. Matter* **31** 453001