Deep Surrogate Models for Atomic Systems

Sathya Chitturi
Department of Materials Science & Engineering
Stanford University

Xiaotong Gui
Department of Statistics
Stanford University

1 Introduction

In this report, we consider the problem of black-box optimization of complex physical systems. We are interested in the scenario where we have a known specification of the system **X** with a corresponding associated value **y**. Given the data the task is to generate a new X^i which yields a desirable value for y^i, defined according to some prior design specification. There has been much recent progress in the inverse materials design space and notable successes include the development of better nanophotonic devices [9] and metasurfaces [12] as well as more promising new pharmaceutical drugs [14]. In this paper, we take a surrogate optimization approach to find optimal input configurations. Specifically, we aim to train deep neural networks to directly learn the **X, y** mapping. The advantages of such an approach is that the resulting neural network is very cheap to evaluate and, in general, the mapping is differentiable. If this neural network can be successfully trained to closely approximate the underlying function, then it should be possible to optimize this new function directly via gradient-based optimization schemes.

We analyze a two-dimensional physical system which consists of a number of atoms which can interact in an attractive or repulsive manner depending on the atomic pairwise distances (Lennard-Jones Potential). Here, we aim to find the configuration of atoms which minimizes the total potential energy. We choose this problem because the functional mapping is known analytically and is fast to evaluate. Our hope is that the analysis presented here can be of use to other materials systems where the function form is unknown and the true forward model is difficult to evaluate.

2 Related work

This work shares many similarities to the well established field of machine learning potential force fields. In this field, there have been a number of successful attempts to learn the potential energy of complex systems [4, 8, 11, 13]; in general, these systems are much more complicated than the Lennard-Jones model treated in this report. The difference between our work and these projects is the information provided to these models during training. For machine learning force fields, the energy for every atom in a configuration is provided to the algorithm. For instance, an atomic cluster with $N_{particles}$ = 32 will have 32 potential energies as the corresponding label. Here, we focus on the situation where only the total energy is known; this is likely a harder problem since the machine learning models have less information to learn from. In our work, we also focus on developing a graph convolutional neural network to predict energies. Our graph implementation is similar to the implementation in [2] which predicts the mobility of particles in a supercooled liquid simulation and the implementation; the primary difference in our work is that we predict energies rather than mobilities.

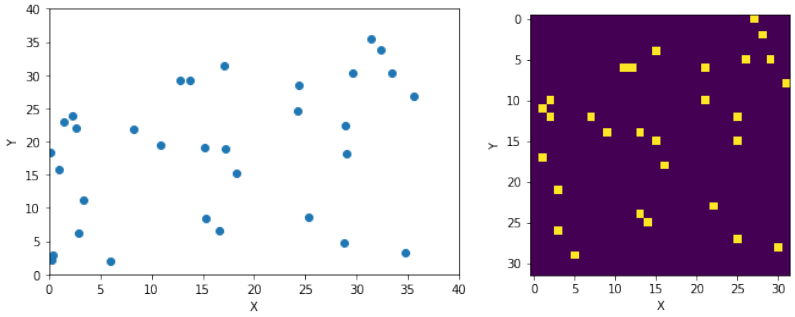
3 Dataset Description

We simulate a dataset of Lennard-Jones Potential which contains the positions of N atoms in 2-dimensions. Specifically, each training example X^i, is represented by a $2 \times N$ matrix corresponding to (x,y) coordinates for each atom. The regression target is the total potential energy (V) of the system (Equation 1); in this equation, r_{ij} is the distance between particle i and particle j and σ and ϵ are

CS230: Deep Learning, Fall 2020, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

physical constants that are set to 1 for our analysis. An example, a scatter plot visualization of a training data example is provided in Figure 1 (a).

$$V = 4\varepsilon \sum_{i}^{N} \sum_{j}^{N} \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right], \quad (1) \tag{1}$$



(a) Example training data; here N = 32 and V = 20.01.

(b) 2D discretization of atomic positions.

Figure 1: Visualization of a single training example for the Lennard-Jones optimization problem.

We simulate train, validation and test set of sizes 50,000, 10,000 and 10,000 respectively (Full Energy Dataset). Where required, we also augmented this dataset by adding 19 permutations of each example to the training set (Section 4.1.1). We also created a second set of training, validation and test sets corresponding to only structures for which $V < 0$ (Low Energy Dataset).

4 Methods and Model Overview

4.1 Analysis of Different Forward Models on the Full Energy Dataset

In this section we elucidate the different architectures and features considered for training a surrogate neural network model to approximate the Lennard-Jones potential. We develop our models in Keras [5] and Tensorflow [1]; we use StellarGraph for the GCN implementation [7]. Here it is worth noting that the energy prediction range for this problem is very large (-7 to 10^{50}). For this reason, we instead opted to transform V to $\log(V + 32)$. Here, the $+32$ factor ensures that only positive numbers are in the logarithm argument. For all analyses in this section, we used a mean-squared error (MSE) loss function between the prediction and $\log(V + 32)$ for training; our evaluation metric was chosen to be the mean-absolute percentage error (MAPE).

4.1.1 Multilayer Perceptron with Permutations

Our first model was a Multilayer Perceptron Regressor (MLP) that is trained directly on the (x, y) coordinates of the simulated data in the Full Energy Dataset (Section 3). Here we note that shuffling the N (x,y) coordinates in the input data gives the same atomic configuration. Thus, we augment our training set by adding 19 extra permutations of each training data example. We tried a number of different fully-connected architectures, varying from 1000-1000000 parameters. In this work, we report the results using a fully-connected architecture with [512, 256, 128, 64, 32, 32, 32, 16, 1] nodes for each hidden layer, ReLU activation functions. This model was trained with a batch size of 512, learning rate of 0.001 for 500 epochs.

4.1.2 Discretization and 2D CNN

Due to the fact that the MLP is not permutation invariant, we next attempted to converting the coordinate matrix information into a 2D image via discretization. We discretized coordinate space into 32 $\times$ 32 small boxes and labelled each box as 1 if it contains a particle. The fineness of discretization was chosen as the smallest box size such that there is no more than 1 particle in each box. Then for each training example, the input data was transformed into a 32 $\times$ 32 binary matrix,

which preserves the permutation invariant property (Figure 1 b). We used a number of different Convolutional Neural Network Regression (CNN) models on this image date. In this work, we report the results using the CNN in Figure 2 on the Full Energy Dataset (Section 3). This model was trained for 200 epochs, with batch size of 64 and a learning rate of 0.0001.

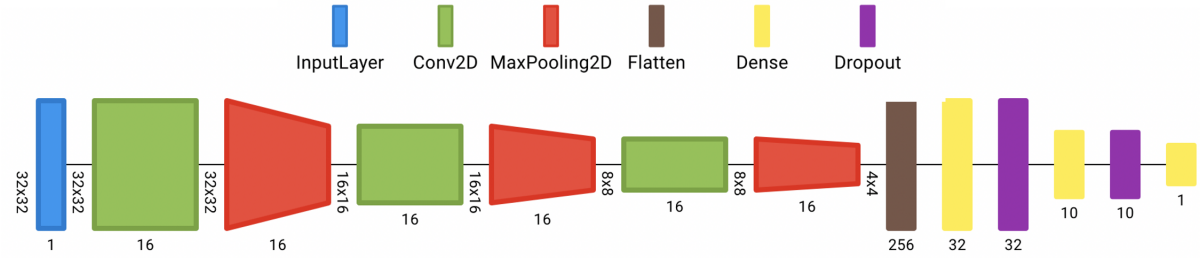


Figure 2: CNN Model Architecture. This illustration was created using Net2Vis [3].

4.1.3 Graph Convolutional Neural Network

Conceptually, graph neural networks is the best candidate for our problem. This is because the functional form of the LJ potential depends only on pairwise-interactions and is dominated by local environments (only nearby particles interact with each other). Furthermore, graph neural networks should be invariant to permutations, translations and rotations of the input data [2].

We converted each training example into an undirected graph where the nodes represent particles and the edges are weighted to represent the distance between particles; two nodes are connected if they are sufficiently close in distance. We chose the distance threshold based on the success of [2] in learning structural information. In addition, we weight the edges using the actual distances.

We created two node features to feed into the encoder:
1. **radial density**: for each node, we calculate the number of surrounding particles within a certain radius using a range of radius from (0.1,32) with a step of 0.5. This metric is a simplified version of kernels proposed in other papers which aim to learn complex atomic environments [15].
2. **sorted pairwise distance**: we pass in the sorted distance of the current particle to every other particle in the system. This feature tells us how far away each node is from everything else.

We then trained a neural network that predicts system energy based on a graph using an architecture similar to the Deep Graph Convolutional Neural Network (DGCN) [16]. The convolutional layers output an embedding for each node which are then passed through a series of fully connected layers to create the final prediction (Figure 3).

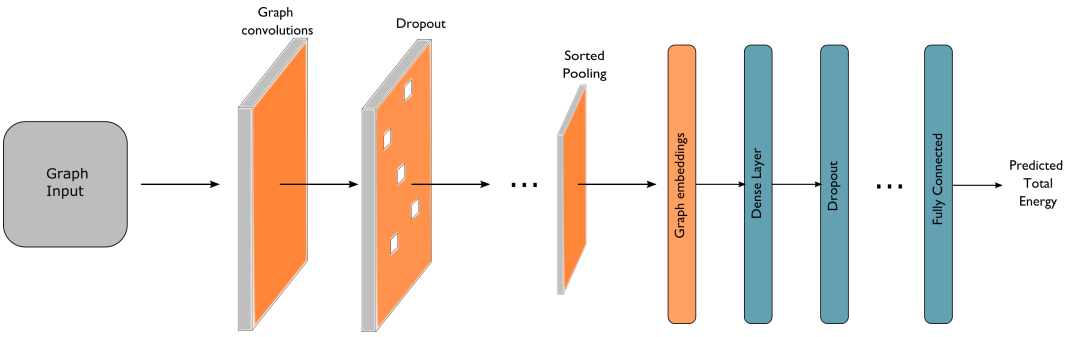


Figure 3: GCN Model Architecture

For this analysis, we used a GCN with n_{Layers} GCN layers each with n_{Nodes} nodes. This is followed by a series of fully connected layers with [128, 64, 32, 20, 1] hidden nodes. All the activation functions were ReLU except for the last linear layer.

4.2 Hyperparameter optimization for GCN models on Low Energy Dataset

We optimized GCN models on the Low Energy Dataset (Section 3) in order to perform well on low energy structures ($V < 0$). Hyperparameter optimization was performed for the GCN models with respect to the learning rate (α), number of nodes in the GCN layers (n_{Nodes}) and number of GCN layers (n_{Layers}). For this dataset, we chose mean absolute error (MAE) for the evaluation metric since the MAPE is very unstable for inputs near 0. Furthermore, since the dataset was restricted to $V < 0$, we trained our models directly on V instead of $\log(V + 32)$. For simplicity, we kept the number of nodes in each GCN layer constant for each model. We performed an exhaustive search of the following hyperparameters: $\alpha \in$ [0.001, 0.0001, 0.00001, 0.000001], $n_{Layers} \in$ [1, 2, 3] and $n_{Nodes} \in$ [32, 128]. Models were trained for 100 epochs with a batch size of 256 and an early-stopping patience of 10.

5 Results and Discussion

5.1 Comparison between MLP, CNN and GCN models

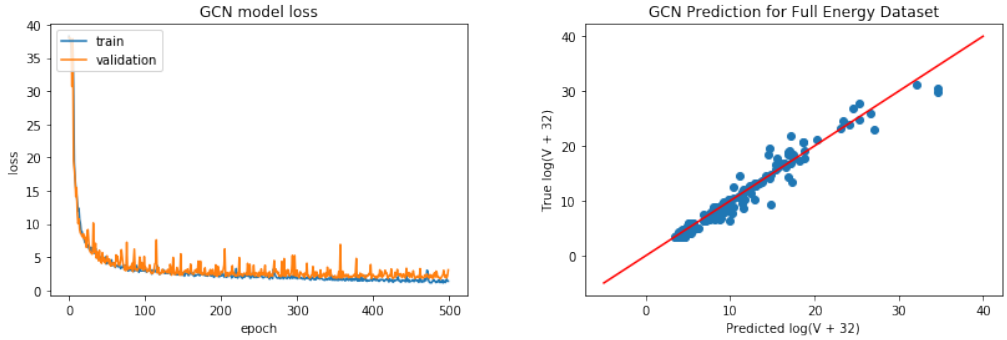
We report the training and test performance for an MLP with permutation, 2D CNN on discretized data and the GCN (Table 1). Please refer to Section 4.1 for a description of the respective architectures.

Model	MAPE (training)	MAPE (validation)
MLP (with permutations)	4.8	78
2D CNN (with discretization)	29	81
GCN	6.3	6.6

Table 1: MAPE of different models on training and validation set

We achieve very poor performance for the permuted MLP and CNN models (Figure 4). The reason for this poor performance is likely because the MLP and CNN are not invariant to rotations of the data features; furthermore, the permuted MLP, is additionally not invariant to translation operations. In addition, it is likely that the sparsity of the CNN representation (i.e. a binary matrix) makes it very difficult to learn meaningful representations. This phenomena has been observed previously in other works [10, 6]. In order to improve our performance using this method, one possible future approach may involve training sparse CNN models in which convolutions are only performed on non-sparse areas of the input [6].

On the other hand, we achieve good results for the GCN implementation (Figure 4). On the validation set, the GCN model achieved good overall performance on systems with total energies ranging from -32 to 10^{40}. We ascribe this improved performance to the fact that GCNs automatically enforce translation, rotational and permutation invariance. From this section it is clear that the architecture choice and featurization is critical to achieve good performance.



(a) MSE Loss for Training and Validation

(b) Predicted and true $\log(V + 32)$

Figure 4: GCN Training Performance

5.2 Error Analysis for GCN trained on Full Energy Dataset

Ultimately, the goal of this work is to find atomic configurations with low energy. We analyzed the performance of our model in Section 5.1 for these low energy structures (Figure 5); here, we convert the potential energies from a log scale back to a linear scale. From this transformation, it is clear that our model performs very badly in this regime. This is relatively unsurprising as the model was trained to be sensitive to logarithmic and not linear changes in energy. For this reason, we decided to additionally train and optimize models to perform well on low energy structures (5.3) using the Low Energy Dataset.

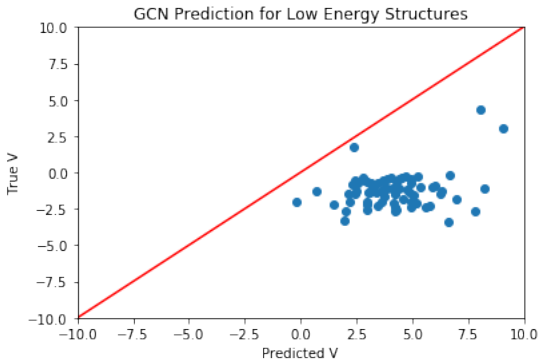


Figure 5: GCN prediction for low energy data.

5.3 Hyperparameter Optimization for GCN trained on Low Energy Dataset

We performed hyperparameter optimization over the learning rate, number of GCN layers and hidden nodes (Section 4.2) in order to find a good model to predict the energy of low energy structures. The top five models for our analysis are shown in Table 2. Here we report the validation MAEs for all models and the corresponding test MAE for the best model.

Model	Learning Rate	n_{Nodes}	n_{Layers}	MAE (validation)	MAE (test)
1	0.001	128	2	0.248	0.244
2	0.001	128	3	0.255	-
3	0.001	32	3	0.305	-
4	0.001	32	1	0.333	-
5	0.0001	128	3	0.343	-

Table 2: Performance of top five GCN models on validation and test set.

In general, our intuition for varying the number of GCN layers was to change the message passing depth of the graph and thus, in this case, to allow for more non-local effects. However, we do not definitively observe better performance for a larger number of layers. We believe that this is because the Lennard-Jones potential is primarily local in nature (energies of interaction are dominated by nearest neighbours) and therefore additional non-local knowledge is relatively inconsequential. In addition, it is worth mentioning that the top five models all perform well and are separated by less than 0.1 MAE. This indicates that GCN model optimization, for this problem, is not as important as selecting an appropriate architecture which correctly reflects the underlying physics (Section 5.1).

6 Conclusion and Future Work

In this work, we trained models to approximate the Lennard-Jones potential. We found that for this problem, the choice of neural network architecture is critical to achieving good performance and that GCNs are useful models for this purpose. We also found that it is difficult to maintain good predictions on both very high and very low energy structures simultaneously. In future work we would like to investigate the effect of using a custom loss function to prioritize prediction on low energy structures. In addition, the main area for further development is to apply black-box optimization techniques to our GCN models in order to find low energy structures.

7 Contributions

Both authors contributed equally to this work. Alex Gui focused on hyperparameter optimization and error analysis and Sathya Chitturi focused on developing different forward model architectures. All portions of the analysis were reviewed by both authors. We thank Sharon Zhou, Peihao Sun, Daniel Ratner and Yanwen Sun for their helpful suggestions and guidance.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Victor Bapst, Thomas Keck, A Grabska-Barwińska, Craig Donner, Ekin Dogus Cubuk, Samuel S Schoenholz, Annette Obika, Alexander WR Nelson, Trevor Back, Demis Hassabis, et al. Unveiling the predictive power of static structure in glassy systems. *Nature Physics*, 16(4):448–454, 2020.

[3] Alex Bäuerle and Timo Ropinski. Net2vis: Transforming deep convolutional networks into publication-ready visualizations. *arXiv preprint arXiv:1902.04394*, 2019.

[4] Venkatesh Botu, Rohit Batra, James Chapman, and Rampi Ramprasad. Machine learning force fields: construction, validation, and outlook. *The Journal of Physical Chemistry C*, 121(1):511–522, 2017.

[5] Francois Chollet et al. Keras, 2015.

[6] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019.

[7] CSIRO's Data61. Stellargraph machine learning library. `https://github.com/stellargraph/stellargraph`, 2018.

[8] Tran Doan Huan, Rohit Batra, James Chapman, Sridevi Krishnan, Lihua Chen, and Rampi Ramprasad. A universal strategy for the creation of machine learning-based atomistic force fields. *NPJ Computational Materials*, 3(1):1–8, 2017.

[9] Tyler W Hughes, Momchil Minkov, Ian AD Williamson, and Shanhui Fan. Adjoint method and inverse design for nonlinear nanophotonic devices. *ACS Photonics*, 5(12):4781–4787, 2018.

[10] Maximilian Jaritz, Raoul De Charette, Emilie Wirbel, Xavier Perrotton, and Fawzi Nashashibi. Sparse and dense data with cnns: Depth completion and semantic segmentation. In *2018 International Conference on 3D Vision (3DV)*, pages 52–60. IEEE, 2018.

[11] Ying Li, Hui Li, Frank C Pickard IV, Badri Narayanan, Fatih G Sen, Maria KY Chan, Subramanian KRS Sankaranarayanan, Bernard R Brooks, and Benoît Roux. Machine learning force field parameters from ab initio data. *Journal of chemical theory and computation*, 13(9):4492–4503, 2017.

[12] Zhaocheng Liu, Dayu Zhu, Sean P Rodrigues, Kyu-Tae Lee, and Wenshan Cai. Generative model for the inverse design of metasurfaces. *Nano letters*, 18(10):6570–6576, 2018.

[13] Rampi Ramprasad, Rohit Batra, Ghanshyam Pilania, Arun Mannodi-Kanakkithodi, and Chiho Kim. Machine learning in materials informatics: recent applications and prospects. *npj Computational Materials*, 3(1):1–13, 2017.

[14] Benjamin Sanchez-Lengeling and Alán Aspuru-Guzik. Inverse molecular design using machine learning: Generative models for matter engineering. *Science*, 361(6400):360–365, 2018.

[15] Michael J Willatt, Félix Musil, and Michele Ceriotti. Atom-density representations for machine learning. *The Journal of chemical physics*, 150(15):154110, 2019.

[16] Muhan Zhang, Zhicheng Cui, M. Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *AAAI*, 2018.