Open Source Infrastructure for Differentiable Density Functional Theory

Advika Vidhyadhiraja [1] **Arun Pa Thiagarajan** [1]
Shang Zhu [2] **Venkat Viswanathan** [2]
Bharath Ramsundar [1]

Abstract

Learning exchange correlation functionals, used in quantum chemistry calculations, from data has become increasingly important in recent years, but training such a functional requires sophisticated software infrastructure. For this reason, we build open source infrastructure to train neural exchange correlation functionals. We aim to standardize the processing pipeline by adapting state-of-the-art techniques from work done by multiple groups. We have open sourced the model in the DeepChem library to provide a platform for additional research on differentiable quantum chemistry methods.

1. Introduction

Density-functional theory (DFT) is used to calculate the electronic structure of atoms, molecules, and solids. Its objective is to use the fundamental laws of quantum mechanics to quantitatively comprehend the properties of materials. There are serious scaling limitations to traditional methods used to approximate solutions to the Schrödinger equation of N interacting electrons moving in an external potential. Whereas in DFT, instead of the high-dimensional many-body wave function, the density $n(r)$ is a function of three spatial coordinates. The many-body electronic ground state can be described using single-particle equations and an effective potential thanks to the Kohn-Sham theory (Kohn & Sham, 1965).

The effective potential is made up of three parts: the exchange-correlation (XC) potential, which accounts for many-body effects, the Hartree potential, which describes the electrostatic electron-electron interaction, and the ionic potential resulting from the atomic cores (Kurth et al., 2005; Pederson & Baruah, 2015). Mathematically, the energy contributors in DFT can be represented as

$$E_{total} = E_{\mathrm{kin}} + E_{\mathrm{el}} + E_{\mathrm{xc}} \tag{1}$$

The kinetic energy term is calculated using a fictitious non-interacting system. The second term captures the electrostatic interactions between electrons and nuclear cores. A potential energy surface is derived using the Born-Oppenheimer approximation to account for the electrostatic repulsion between the nuclear cores (Voss, 2022). The most commonly used and simplest class of XC functionals are the local-density approximations (LDA) which mandate that E_{xc} depends only on $n(r)$ and not on its derivatives. Functionals such as the LDA class are traditional approximate forms derived by humans and are widely used due to their accuracy(Parr & Yang, 1995) .

1.1. DeepChem and Differentiable Physics

DeepChem is an open source python library for scientific machine learning and deep learning on molecular and quantum datasets (Ramsundar et al., 2021a). Deepchem provides a framework to solve difficult scientific problems in areas such as drug discovery, energy calculations and biotech (Wu et al., 2018). It does so by specifying that scientific calculations must be broken down into workflows built out of underlying primitives such as data loaders, featurizers, data splitters, learned models, metrics and hyperparameter tuners. This systematic design allows DeepChem to be applicable to a wide variety of applications. For example, DeepChem has enabled large scale benchmarking for molecular machine learning through the MoleculeNet benchmark suite (Wu et al., 2018), protein-ligand interaction modeling (Gomes et al., 2017), generative modeling of molecules (Frey et al., 2022), and more.

DeepChem aims to support open source differentiable programming infrastructure for scientific machine learning, but this effort is a work in progress (Ramsundar et al., 2021b). This research program is broadly known as differentiable physics (Ramsundar et al., 2021b). An important application of differentiable physics is to use neural architectures to accelerate the solution of differential equations. In this work, we aim to solve the self-consistent Kohn Sham equations to calculate electron density $n(r)$. Typically, solution methods for self-consistent calculations

[1]Deep Forest Sciences, California, USA [2]Department of Mechanical Engineering, Carnegie Mellon University, Pennsylvania, USA. Correspondence to: Bharath Ramsundar <bharath@deepforestsci.com>.

Accepted after peer-review at the 1st workshop on Synergy of Scientific and Machine Learning Modeling, SynS & ML ICML, Honolulu, Hawaii, USA. July, 2023. Copyright 2023 by the author(s).

can be slow. Differentiable techniques enable rapid calculations by introducing rich neural approximation schemes.

1.2. Differentiable DFT Methods

The incorporation of machine learning methods into DFT has been going on for over a decade (Pederson et al., 2022). Our model has been derived from XCNN (Kasim & Vinko, 2021). In this method, the xc-functional is learned using a deep neural network and hybridized with a traditional xc-functional. One of the biggest strengths of this approach, is that it is generalizable to a wide range of systems since the neural architecture does not depend on a special physical system (Kasim & Vinko, 2021). Similarly, another advance in this field was made by the DM21 functional (Kirkpatrick et al., 2021), which was computed using fictional systems having fractional charge and spin constraints in order to avoid errors encountered by traditional xc-functionals. These errors are observed for charge densities involving mobile charges and spins(Kirkpatrick et al., 2021).

1.3. Standardizing Differentiable DFT Workflows

We aim to standardize (”deepchemize” (Ramsundar et al., 2021a)) the computation performed by the differentiable DFT model, by implementing the model using the workflow structure implemented by other DeepChem models. To verify correctness, we run experiments and compare results with the original XCNN model (Kasim & Vinko, 2021). By standardizing the process of training a neural network exchange correlation functional, we aim to make it simpler to experiment with new differentiable DFT architectures and enable larger systematic computations.

XCModel (the XCNN implementation in DeepChem), can be used with different loss functions, metrics and machine learned models present in DeepChem, providing a flexible framework for xc-functional design. This flexibility of the XCModel can be used to incorporate more global density information to approximate a more accurate exchange correlation functional. For example, we are currently planning on introducing fractional constraints similar to the DM21 functional to the XCModel. This project would yield a functional that is a combination of DM21 and XCNN. The functional would be trained using a neural network and a fully differentiable quantum chemistry library while also being trained on data points that contain fictional systems with fractional charges and spins.

```
- e_type : 'ae'
  true_val : '0.09194410469'
  systems : [{
        'moldesc': 'Li 1.5070 0 0; H -1.5070 0 0',
        'basis': '6-311++G(3df,3pd)'
    }, {
        'moldesc': 'Li 0 0 0',
        'basis': '6-311++G(3df,3pd)',
        'spin': 1
    }, {
        'moldesc': 'H 0 0 0',
        'basis': '6-311++G(3df,3pd)',
        'spin': 1
    }]
```

Figure 1. Training data is provided in a Yaml format as displayed above. Input data is processed with the DFTYamlLoader class.

2. Implementation

2.1. Dataset

There are four types of data points used to train XCModel: atomization energy (AE) calculations, ionization potential (IP) calculations, density profile regulations and density matrix calculations. The ground truth values for the first two types are obtained from NIST databases (Kasim & Vinko, 2021), and the rest are calculated by performing CCSD calculations (using PYSCF (Sun et al., 2018)). Users do not need to enter the equations used to calculate the total energy for AE and IP data points. In (Kasim & Vinko, 2021), the training dataset consists of all four types of data points. However, we have only used AE, IP and Density matrix calculations thus far. We have implemented a DFTYamlLoaderclass in DeepChem that loads and prepares data, and featurizes the data into standard molecular objects. We use a specific format to build the yaml file. Each molecule is known as an “entry” object , and contains multiple “system” objects. Each system contains information on the molecule description, basis set, charge, and spin number. The true values are either numbers or .npy files. For predictions, the true values do not need to be entered. An example of a data point can be seen in Figure 1

2.2. Layers

A layer can be defined as a function that transforms a tensor into another. The first layer in the XCModel is the Neural Network XC (NNXC) layer, where the exchange correlation functional is trained based on a pre-defined traditional class of functionals. Currently we have implemented an NNLDA and NNPBE layer. The electron and spin densities are transformed to be the input of the neural network. We plan to implement Meta-GGA based layers in the future. The output from this layer is used in the HybridXC layer, which computes XC energy by summing XC energy computed from libxc (with any conventional DFT functional) and the trainable neural network with tunable weights. This layers corresponds to equation 2 from (Kasim & Vinko, 2021)

$$E_{nnLDA}[n] = \alpha E_{\mathrm{LDA(xc)}}[n] + \beta \int n(r) f(n,\varepsilon) dr \quad (2)$$

The output from the HybridXC layer is used to calculate the XC potential which is used to solve the self consistent iterations. This process is carried out in the SCF (Self Consistent Field) layer. The SCF iterations are done using DQC's fully differentiable KS modules (Kasim & Sofroniew, 2021). This layer can also be used to perform dft calculations without any NNXC, i.e, any traditional xc functional. The outputs from this layer are the electron densities. The gradient propagation of the self-consistency cycle is done using an implicit gradient calculation instead of linear mixing using xitorch (Kasim & Vinko, 2021). These layers are used in the XCModel. An ordinary feed forward neural network (implemented in PyTorch) can be used to train the XC functional. The torch model can also be used with various loss functions and metrics present in DeepChem . In our experiments, we use a L^2 loss to train the model and mean absolute error (MAE) as the metric. A schematic of the implementation can be seen in Figure 2.

3. Results and Observations

Table 1 is a comparison between MAE values for two different test datasets; when computed using various trained/ traditional functionals. The test datasets we use are: ionization potential for the atoms H-Ar, and atomization energy for 16 Hydrocarbons. The exact molecules can be found in the supplementary material of (Kasim & Vinko, 2021). Results indicate that the DeepChem implementation slightly trails the performance of the original implementation from (Kasim & Vinko, 2021), but we anticipate that this gap will close once we complete the implementation. The training and testing for these experiments can be run on a 16GB RAM CPU system. However, for testing on larger molecules and datasets, a GPU system would likely be required.

Calculations	IP 18	AE 16 HC
LDA	24.6	48.7
XCNN-LDA-IP	15.2	25.4
DC-XCNN-LDA-IP	24.2	28.8

Table 1. MAE scores in kcal/mol for two test datasets

3.1. Hydrogen Dissociation

In Fig 3, we plot the dissociation energy for Hydrogen at different points and compare the values with CCSD values. The computed dissociation curve computed by XCModel closely tracks the exact values.

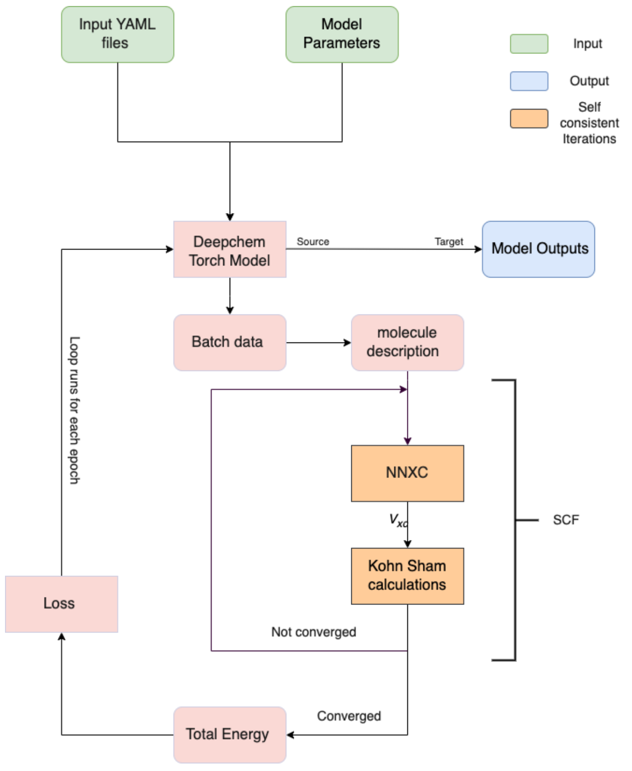

Figure 2. Schematics of the exchange correlation XCModel. The yaml files are loaded using the DFTYamlLoader and featurized into mol objects using the DFT data classes. The model parameters consist of the PyTorch model used to train the functional and the choice of loss function. The forward method initializes the Neural Network LDA layer (NNXC), and hybridizes the functional with a traditional LDA functional. The hybrid xc is used to solve the Kohn-Sham equations. Once the self consistent iterations converge, the total energy of the data point is calculated and used to calculate the loss. The trained model can be used with DeepChem functions such as evaluate and predict with different metrics (Kasim & Vinko, 2021; Ramsundar et al., 2021a)
.

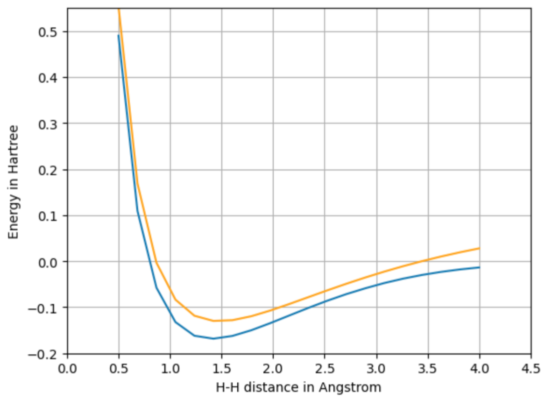

Figure 3. Hydrogen Dissociation Curve. The orange graph represents the exact values(Kasim, 2021) of the energy required for H2 to dissociate at different bond lengths. The blue graph represents the predicted energies using the trained model. We have used 20 data points for this calculation.

4. Conclusion

From Table 1, we observe that the MAE deviations from our model are slightly more accurate compared to a traditional LDA functional, despite not being as accurate as XCNN (Kasim & Vinko, 2021). We can also conclude from the hydrogen dissociation diagram that predicted values are quite close in value to the exact values. Since XCModel is flexible with its parameters, we have also experimented with using various loss functions, metrics, and PyTorch models to train the functional. In the near future, we plan on implementing a few features to further increase accuracy of the DFT calculations while also making the model more versatile. These features include fractional constraints on fictional systems, and layers based on PBE and Meta-PBE. We anticipate that the open DeepChem implementation will enable additional rapid experimentation with differentiable xc-functional architectures.

5. Impact Statement

Differentiable DFT software infrastructure could enable systematic construction of more accurate exchange correlation functions, enabling DFT calculations to have greater impact in materials design and biotechnology applications where existing functionals lack accuracy today.

References

Frey, N. C., Gadepally, V., and Ramsundar, B. Fastflows: Flow-based models for molecular graph generation. *arXiv preprint arXiv:2201.12419*, 2022.

Gomes, J., Ramsundar, B., Feinberg, E. N., and Pande, V. S. Atomic convolutional networks for predicting protein-ligand binding affinity. *arXiv preprint arXiv:1703.10603*, 2017.

Kasim, M. Xcnn. `https://github.com/mfkasim1/xcnn`, 2021.

Kasim, M. and Sofroniew, N. Diffqc/dqc. `https://github.com/diffqc/dqc`, 2021.

Kasim, M. and Vinko, S. Learning the exchange-correlation functional from nature with fully differentiable density functional theory. *Physical Review Letters*, 127(12), sep 2021. doi: 10.1103/physrevlett.127.126403. URL `https://doi.org/10.1103%2Fphysrevlett.127.126403`.

Kirkpatrick, J., McMorrow, B., Turban, D. H. P., Gaunt, A. L., Spencer, J. S., Matthews, A. G. D. G., Obika, A., Thiry, L., Fortunato, M., Pfau, D., Castellanos, L. R., Petersen, S., Nelson, A. W. R., Kohli, P., Mori-Sánchez, P., Hassabis, D., and Cohen, A. J. Pushing the frontiers of density functionals by solving the fractional electron problem. *Science*, 374(6573):1385–1389, 2021. doi: 10.1126/science.abj6511. URL `https://www.science.org/doi/abs/10.1126/science.abj6511`.

Kohn, W. and Sham, L. J. Self-consistent equations including exchange and correlation effects. *Phys. Rev.*, 140: A1133–A1138, Nov 1965. doi: 10.1103/PhysRev.140.A1133. URL `https://link.aps.org/doi/10.1103/PhysRev.140.A1133`.

Kurth, S., Marques, M., and Gross, E. Density-functional theory. pp. 395–402, 2005. doi: https://doi.org/10.1016/B0-12-369401-9/00445-9. URL `https://www.sciencedirect.com/science/article/pii/B0123694019004459`.

Parr, R. G. and Yang, W. Density-functional theory of the electronic structure of molecules. *Annual Review of Physical Chemistry*, 46(1):701–728, 1995. doi: 10.1146/annurev.pc.46.100195.003413. URL `https://doi.org/10.1146/annurev.pc.46.100195.003413`. PMID: 24341393.

Pederson, M. R. and Baruah, T. Chapter eight - self-interaction corrections within the fermi-orbital-based formalism. 64:153–180, 2015. ISSN 1049-250X. doi: https://doi.org/10.1016/bs.aamop.2015.06.005. URL `https://www.sciencedirect.com/science/article/pii/S1049250X15000087`.

Pederson, R., Kalita, B., and Burke, K. Machine learning and density functional theory. *Nature Reviews Physics*, 4(6):357–358, May 2022. doi: 10.1038/s42254-022-00470-2.

Ramsundar, Peastman, Amacbride, and nvtrang91. Making deepchem a better framework for ai-driven science, Apr 2021a. URL `https://forum.deepchem.io/t/making-deepchem-a-better-framework-for-ai-driven-science/431`.

Ramsundar, B., Krishnamurthy, D., and Viswanathan, V. Differentiable physics: A position piece, 2021b.

Sun, Q., Berkelbach, T. C., Blunt, N. S., Booth, G. H., Guo, S., Li, Z., Liu, J., McClain, J. D., Sayfutyarova, E. R., Sharma, S., et al. Pyscf: the python-based simulations of chemistry framework. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 8(1):e1340, 2018.

Voss, J. Exchange-correlation functionals, 2022. URL `https://stanford.edu/~vossj/slac/project/xc-functionals/`.

Wu, Z., Ramsundar, B., Feinberg, E., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. Moleculenet: a benchmark for molecular machine learning. *Chem. Sci.*, 9:513–530, 2018. doi: 10.1039/C7SC02664A. URL `http://dx.doi.org/10.1039/C7SC02664A`.