Machine Learning From Quantum Chemistry

Muawiz Chaudhary[1]; Jialin Liu[2]; Paul Sinz[2]; Yue Qi[2]; Matthew Hirn[2]
[1]Western Washington University, [2]Michigan State University

Advanced Computational Research Experience for Students

Introduction

What is electron density?
Electron density is a measure of the probability of an electron being present at a specific location[1]. Knowing the electron density gives us information on the bonding of atoms, which give rise to advances in drug discovery and batteries. Current methods for calculating electron density are expensive, with the best methods using density functional theory (DFT) being $O(N^3)$.

How can machine learning (ML) help?
Given a set of input output pairs, a ML algorithm can learn to produce output that resembles output from the original data set distribution. ML holds the promise for bypassing the need to do expensive DFT computations.

What is the current state of the art?
- Brockherde and others[2] obtained electron density by learning Hohenberg Kohn mapping from gaussian potentials to electron density.
- Their method used a kernel ridge regression (KRR) model for learning. The accuracy of KRR modes is sensitive to the selection of training data. For kernel method, there is need for a large number of weights in order to capture nonlinearities.
- Related: Isola and others presented idea of “given an outline, fill it in” in their paper Image-to-Image Translation with Conditional Adversarial Networks.[3]
- Deep learning methods have been successful with capturing nonlinearity, with the most popular deep learning architectures being **convolutional neural networks**.

Methods

Generate the dataset

We use ab initio methods to calculate the electron density for 625 different configurations of a Lithium-Oxygen-Lithium system. These electron densities are our **labels**.

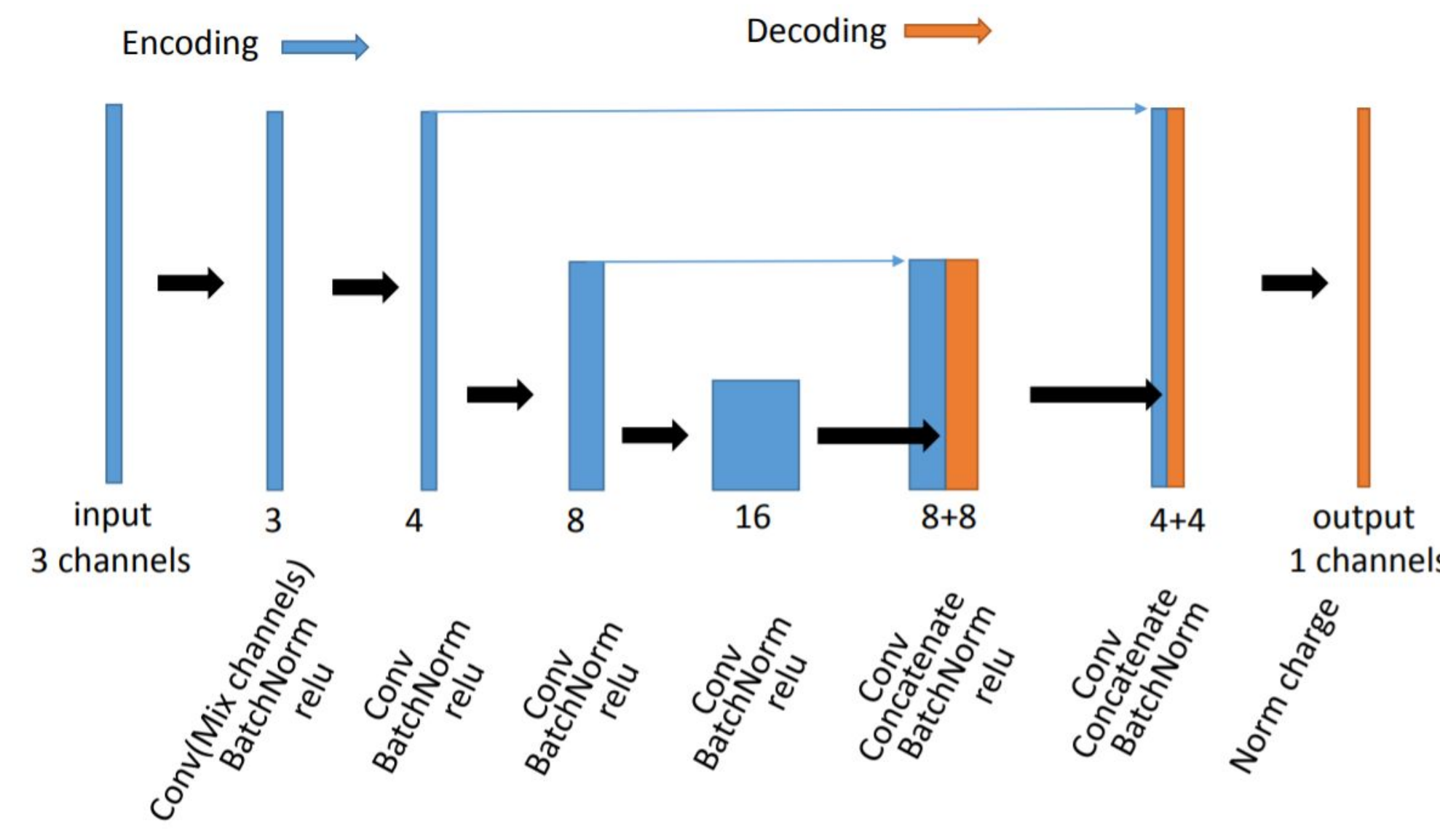
From the same calculations, we look at the atoms and their positions, and model the electron density for each atom using gaussians. The resulting matrix of electron densities are our **inputs**.

Choose a model

Inspired by pix2pix, we choose a **Unet** architecture. We use L2 norm loss function and batch normalization layers.

Consists of a contracting path and an upsampling path. Every layer but the last performs a series of convolutions, batch norm, then a non linear activation function called Rectified Linear Unit (ReLU).

We use Unet implementation by Akeret and others[4].



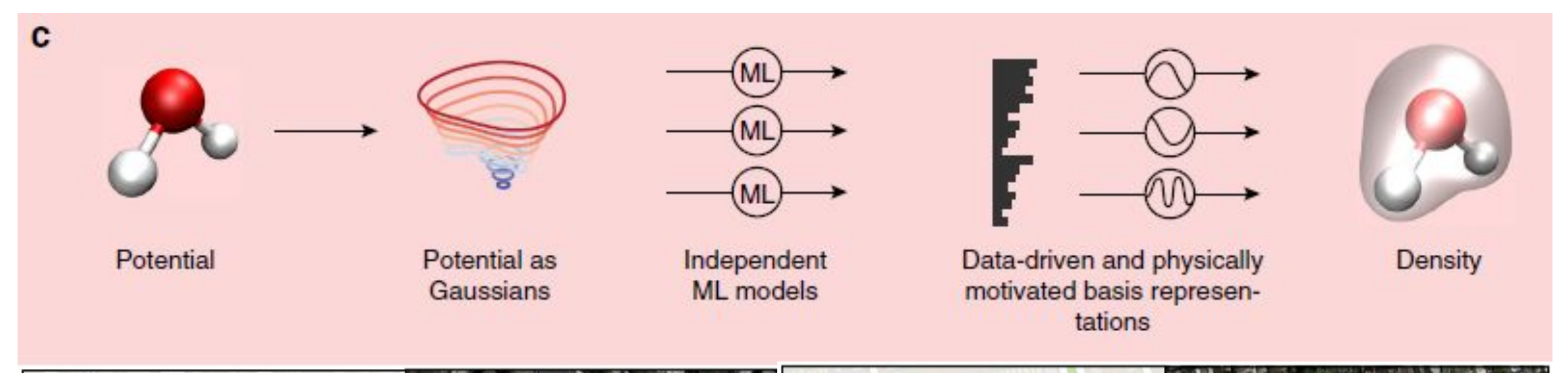
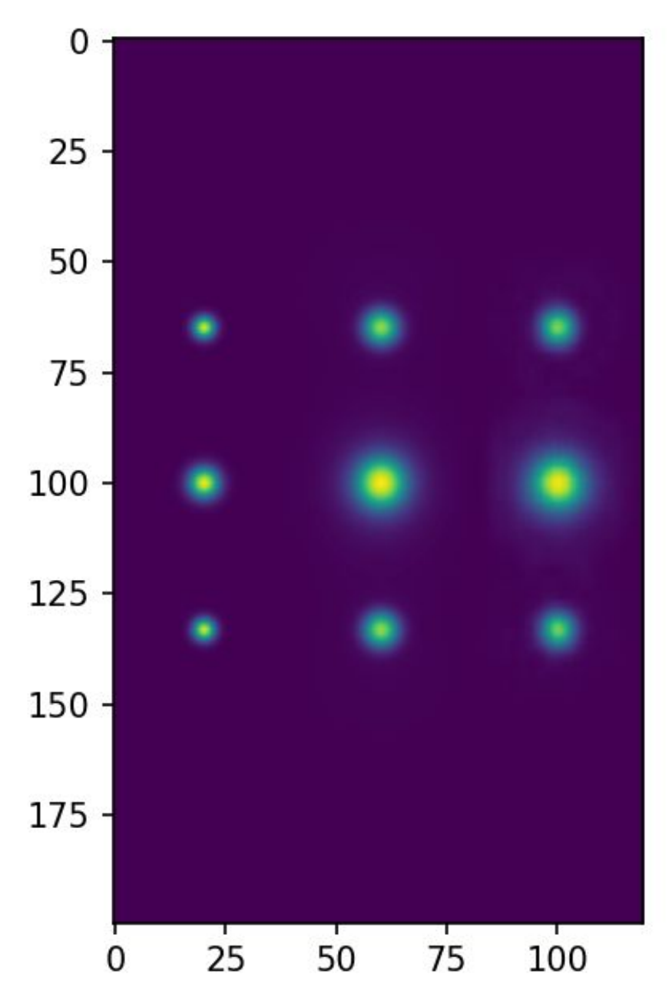


Image on left: view of our Unet architecture. We feed images through a mixing layer, which then passes the images through a series of encoders and decoders. Each encoder applies a convolution, batchnorm, ReLU, then doing a max pooling before being sent off to the next encoder. Each decoding layer does a similar operation to what encoders do, but instead of a max pooling, we perform an upconvolution. Image on top right: Outlines the methodology of Brockherde and others. They modeled potential as gaussians, and fed these as inputs to their KRR model. Their model then chose physically motivated basis representations based off of the data it learned from. Finally, from the representations, we get electron density. Image from “Bypassing the Kohn Sham equations via machine learning”. Images on bottom right: The images on the left are from Google street view, and are inputs to the pix2pic model. The model outputs the images on the right, ‘filling’ in the gaps. Images attributed to Isola and others.

Run and evaluate experiments

Inputs (electron density modeled as gaussians) are fed into a mixer, which then feeds into the model for training. Each input has a corresponding label. After every training iteration, the model makes a **prediction** and compares the prediction against the label. Weights of the model are updated in order to reduce the loss.

After training is finished we evaluate our model against a test set with electron densities the model did not encounter during training. We calculate a average loss over all points in the test set. The goal of **hyperparameter tuning** is to reduce this average loss.



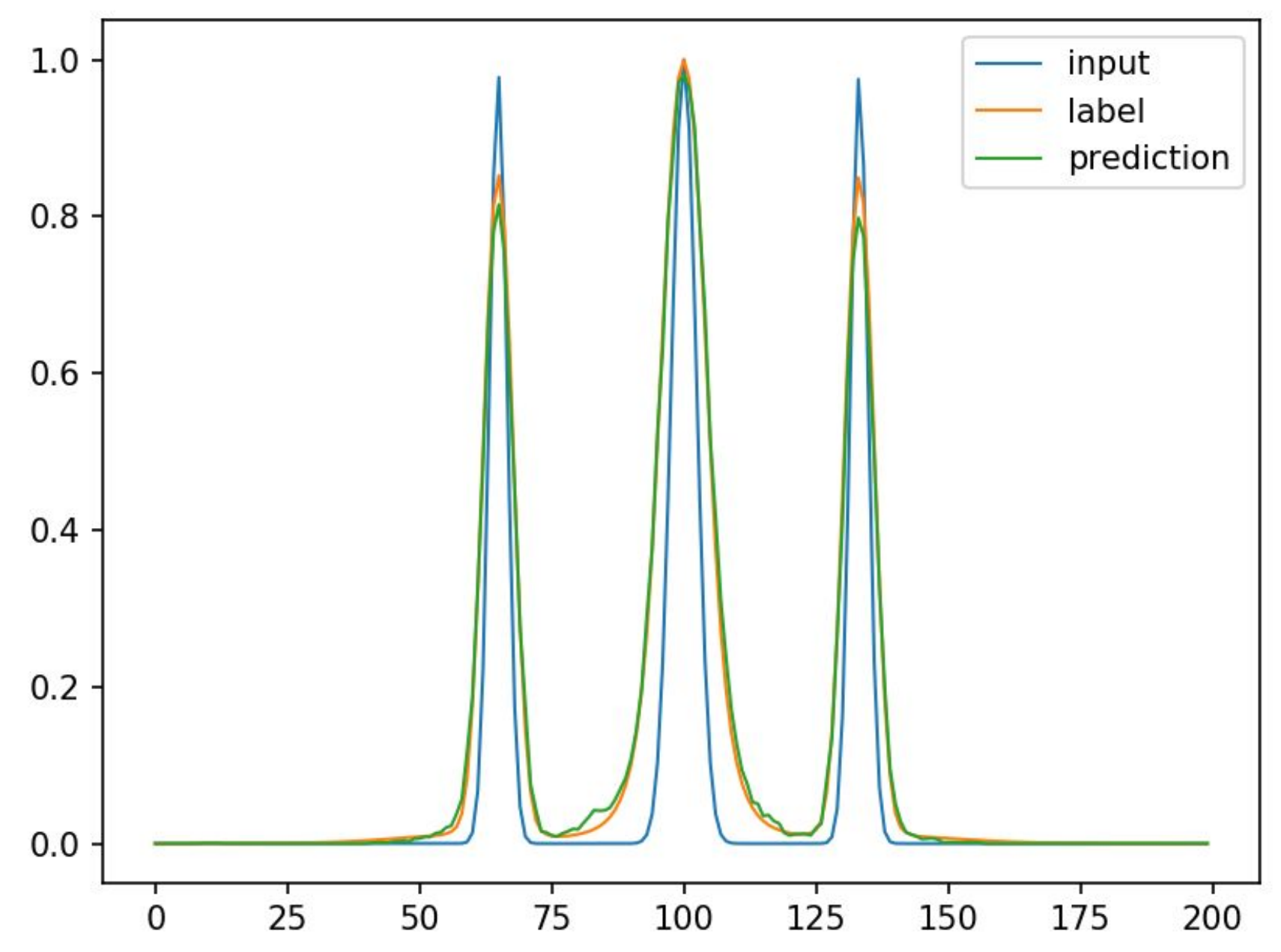


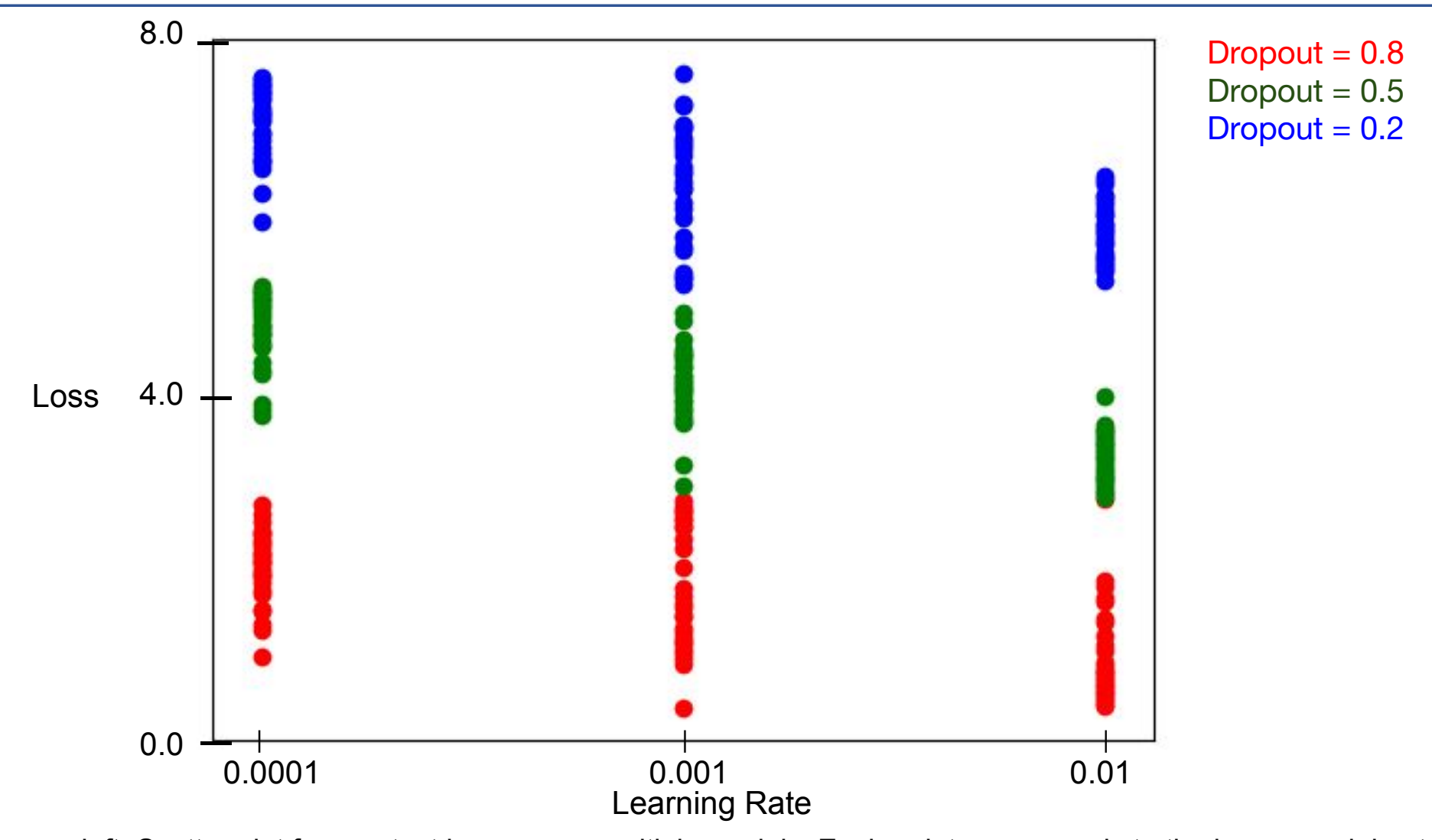
Image on left: View of our electron density. Electron densities are as ordered: Left is the input, middle is the label, and right is the predicted electron density. Image on right: Slice through each electron density. Blue is our input, orange is our label, and green is our prediction.

Results

Our models generate predictions that are very close to our labels. Many predictions, however, come with some extra ‘dust’ of electron density in between atoms.

Additionally, as evidenced by filters that our model outputs, the model learns to differentiate between lithium and oxygen atoms without being told what the atoms are. That is, the model appears to be learning the underlying chemistry of our system.

As a result of hyperparameter tuning using a parallelized grid search, it was realized that dropping dropout led to increased performance on the unet architecture. This observation supports existing literature[5] which recommends the removal of dropout when using batch normalization.



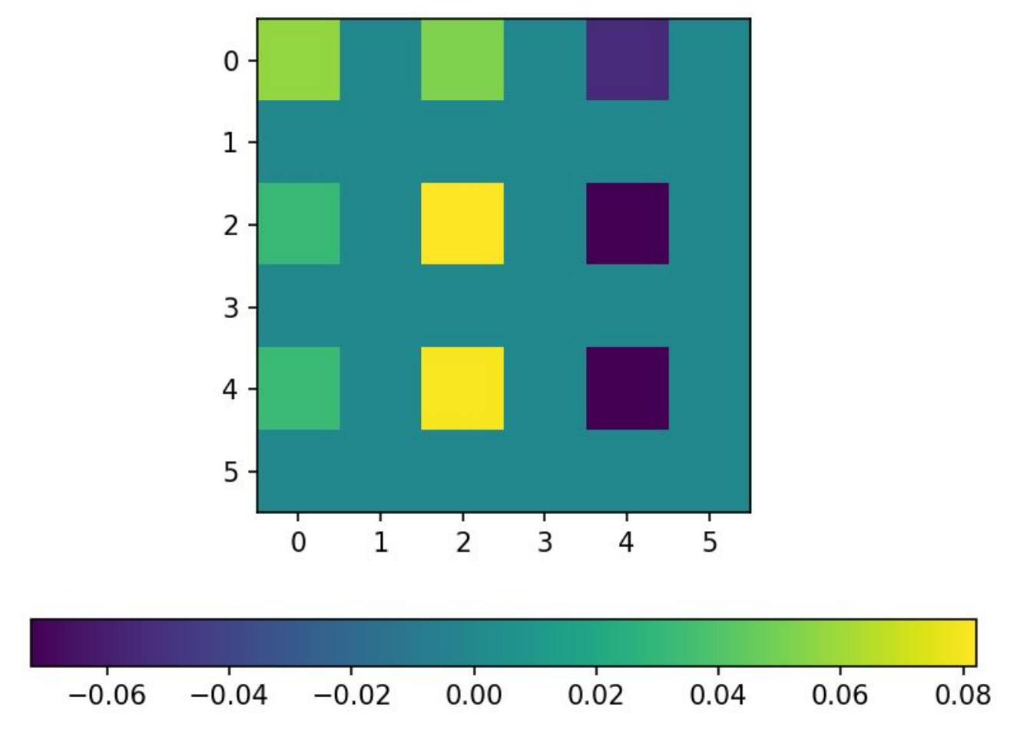


Image on left: Scatter plot for one test image over multiple models. Each point corresponds to the loss a model outputs for the test image. We see that larger dropout and learning rate gives us lower loss. Image on right: Filter that our machine learning model is learning. Produced 26 epochs into training. During training we pass a 3 channel electron density modeled as gaussians into our model. The first channel contains modeled electron density for the oxygen atom, second and third channel contains the modeled electron density for each lithium.

Conclusions

Future research endeavors will be focused towards the application of tools from optimal transport. Long term goals are to see if a model trained on multiple sets of configurations of atoms can correctly predict electron density for a new and larger system that the model has never encountered before.

We would like to acknowledge the support of the NSF through the grant ACI-1560168.

Contact

Muawiz Chaudhary
Western Washington University
Chaudhm@wwu.edu
925-448-6248

References

1. Wikipedia
2. By-passing the Kohn-Sham equations with machine learning by Felix Brockherde and others.
3. Image-to-Image Translation with Conditional Adversarial Networks by Phillip Isola and others
4. Radio frequency interference mitigation using deep convolutional neural networks by Joel Akeret and others
5. Understanding the Disharmony between Dropout and Batch Normalization by Variance Shift By Xiang Li and others.