BandGap Prediction Using a 3D boxel Approach: *A proof of concept*

FRANZ RIVERA TELLEZ

Compiled March 29, 2023

COMING SOON

1. INTRODUCTION

Band-gap is a fundamental property of materials in which the energy difference between two energy bands, the valence band and the conduction band, determines their electronic and optical properties. It refers to the minimum amount of energy that is required for an electron to move from the valence band to the conduction band, thus becoming free to conduct electricity. The size of the band gap determines the conductivity of a material, with wider band gaps corresponding to insulating materials and narrower band gaps corresponding to conductive materials.

Machine learning, a powerful tool in modern research, has been increasingly applied in chemistry and materials science to accelerate the discovery and design of new materials. With machine learning techniques, scientists can predict the properties of materials before they are synthesized, saving time and resources. Machine learning can also be used to optimize the synthesis routes of new materials, reducing trial and error experiments. The predictions made by machine learning algorithms can guide the design of new materials with specific properties, enabling researchers to tailor materials for specific applications.

A. DFT Hybrid method

Density functional theory (DFT) has been widely used by researchers for computational materials modeling, with a particular focus on predicting band gaps in solid-state materials. Accurately predicting band gaps has remained a difficult task. To tackle this challenge, Wing and colleagues suggest a new method that combines hybrid functionals with a separation of interelectronic Coulomb repulsion and an optimized screening parameter determined by a localized orbital framework. This new approach enhances the precision of band gap predictions for solid-state materials.

B. Graph neural networks

GNNs are a class of machine learning models that operate on graph-structured data and have strong ties to the field of geometric deep learning. These models are closely linked to the field of geometric deep learning and are highly applicable to domains such as chemistry and materials science. GNNs can learn about the relationships between the atoms and molecules in the substance, which can help in predicting various properties such as reactivity, stability, and conductivity. Due to their ability to work with graph-structured data, as describe in "Graph neural networks for materials science and chemistry".

C. Vision-based

This method uses convolutional neural networks (CNN's) which use the projected images of 3D spatial models of molecules as inputs (2D Images of the molecules). The CNN's are trained using a dataset of molecular graphs along with their corresponding electronic band-gap values as calculated by conventional methods.

2. METHODOLOGY

A. 3D CNN

In a similar way as the computer vision-based technique proposed by paper "A 3D orthogonal vision-based band-gap prediction using deep learning: A proof of concept" was used, in this the feasibility of using a 3d model represented by boxes with a size of 128x128x128x3 will be explored together with a 3D CNN to make a regression and obtain an approximate value from the band-gap of an organic molecule.

B. Data set description

The data used in this paper comes from the OMDB-GAP1 database, which is a dataset of 12,500 organic molecules, where each molecule is made up of multiple atoms in the XYZ coordinate format along with the bandgap value corresponding to each molecule in a separate file.

B.1. Data collection and data transformation

For the experiments, 2 approaches will be used, the first consists of a direct transformation of the XYZ coordinates of the molecules to a boxel format, while the other consisted of generating the Boxels from NERF (Neural Radiance Fields).

B.2. 3D Boxels direct generation

1. Defined an arbitrary color for each atom type and normalized from zero to one to make training faster.

2. Create a data table with atomic radii of each atom, as well as its atomic number to define the size of each atom, the smallest being hydrogen with a radius of 1.

3. The coordinates of each atom are extracted and then normalized to be able to put them in a vector space of 128x128x128 and 3 channels for RGB.

4. All empty space is placed with a value of -1 which indicates that there are no atoms.

5. The spheres of the atoms are created with their corresponding radius and with their new normalized position in space using to create the boxels representation is used the following formula.

6. The connections between atoms are created using the atomic radius multiplied by an arbitrary scalar. In this case it was decided to use 1.5. To archive this the next steps are followed:

   (a) Make a vector v, which is the vector of the bond and then divide it by its magnitude.

   (b) Make one vector that is not in the same direction as v.

   (c) Make a new vector n1 which is perpendicular to v and then normalize it.

   (d) Make unit vector n2 perpendicular to v and n1.

   (e) Create meshgrid from 0 to 2Π for the bond vector and the two vectors calculated in the previous steps.

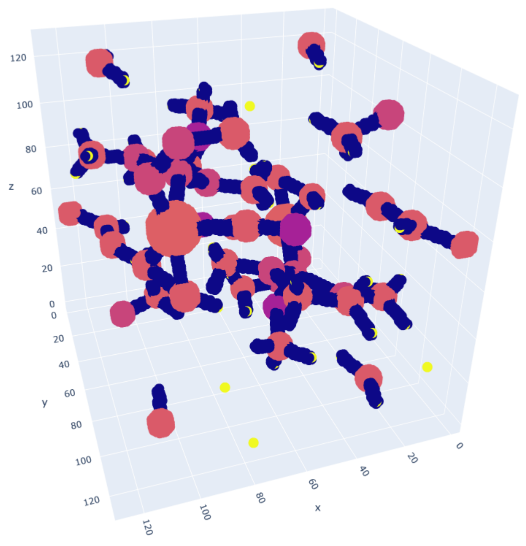
   (f) Create he coordinates for the bond cylinder.

$$X, Y, Z = p + v * t + R * sin(\theta) * n1 + R * cos(\theta) * n2 \tag{1}$$

   (g) Remove rows with nan values from the coordinates.

   (h) Loop through all the points in the cylinder and set the color of the boxel in the world.

Fig. 1. Boxels direct generated (not real color)



B.3. Images to NERF to boxels

1. To generate a 2D representation of the molecules, we used the mogli library, which provides us with a gr3 interface to go from an XYZ coordinate format file of the atoms to 3D molecules and then export them to an image format.

2. Generate the transformation for camera matrix, for that the next formula is used:

$$M = TR = \begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r^{11} & r^{21} & r^{31} & 0 \\ r^{12} & r^{22} & r^{32} & 0 \\ r^{13} & r^{23} & r^{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} r^{11} & r^{21} & r^{31} & \Delta x \\ r^{12} & r^{22} & r^{32} & \Delta y \\ r^{13} & r^{23} & r^{33} & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$(TR)^{-1} = R^{-1}T^{-1} \tag{3}$$

$$\mathbf{R} \text{ is orthogonal} \Leftrightarrow \mathbf{R}^{-1} = \mathbf{R}^{T} \tag{4}$$

$$\mathbf{R}^{T} = \begin{bmatrix} r^{11} & r^{21} & r^{31} & 0 \\ r^{12} & r^{22} & r^{32} & 0 \\ r^{13} & r^{23} & r^{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{T} = \begin{bmatrix} r^{11} & r^{12} & r^{13} & 0 \\ r^{21} & r^{22} & r^{23} & 0 \\ r^{23} & r^{32} & r^{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

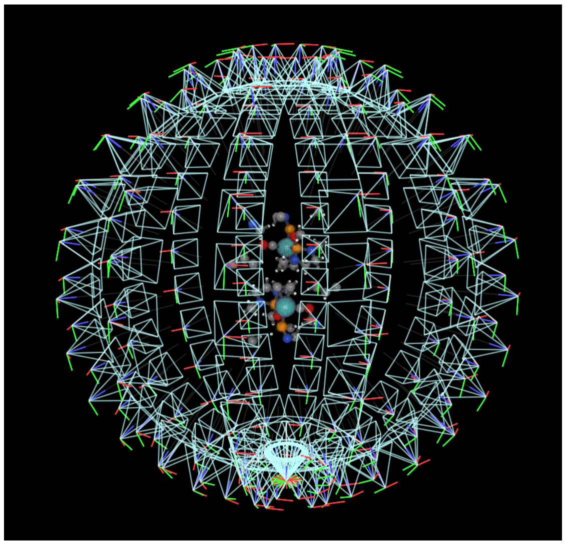
$$\mathbf{T}^{-1} = \begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 0 & 0 & -\Delta x \\ 0 & 1 & 0 & -\Delta y \\ 0 & 0 & 1 & -\Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

$$n = x\text{-}axis, u = y\text{-}axis, v = z\text{-}axis, e = eye \tag{7}$$

$$\begin{aligned} M^{-1} &= (TR)^{-1} \\ &= R^{-1}T^{-1} \\ &= \begin{bmatrix} n_x & u_x & v_x & 0 \\ n_y & u_y & v_y & 0 \\ n_z & u_z & v_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 & e_x \\ 0 & 1 & 0 & e_y \\ 0 & 0 & 1 & e_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \\ &= \begin{bmatrix} n_x & n_y & n_z & 0 \\ u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -e_x \\ 0 & 1 & 0 & -e_y \\ 0 & 0 & 1 & -e_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} n_x & n_y & n_z & -(n \cdot e) \\ u_x & u_y & u_z & -(u \cdot e) \\ v_x & v_y & v_z & -(v \cdot e) \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{8}$$

3. 200 images were generated along with the transformation matrix of the camera used to generate each of these images this was the input given to nerf.
4. To facilitate the training and faster generation of the data set, the Nvidia nerf implementation called instant-ngp was used with slight modifications to obtain an 128x128x128x4 output and to be able to use it through the terminal and not through the graphical interface.
5. Finally, the NERF output was modified to have only 3 channels by removing the alpha channel.

Fig. 2. Boxels generated with NERF and Cameras



3. MODEL ARCHITECTURE

It was decided to use 2 models to obtain the bangap, both are based on the use of convolutional neural networks (CNN) with slight differences between them.

A. Simple CNN

This network consists of a 3D CNN formed by 6 convolution blocks and another 5 blocks for regression, each convolution block is formed by a Conv3d, a LeakyReLU activation card and a MaxPool3d layer, Each conclusion block is formed by a layer FC a LeakyReLU a BatchNorm1d and a Dropout with a value of .2

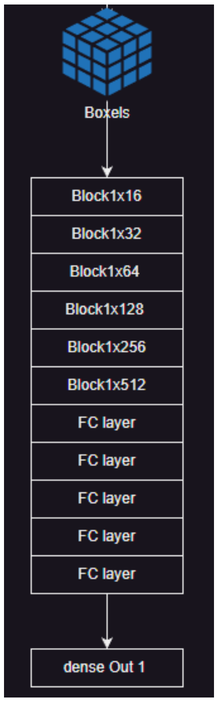
B. CNN + XYZ

This model follows the same structure of the simple CNN but before starting the regression blocks, they concatenate the data of the coordinates of the atoms in XYZ format to give more exact information about the position of each atom since the reduction to a space of 128x128x128 makes decimal values unable to be represented correctly.

4. TRAINING NETWORKS

For the training of these neural networks, it was decided to use the Adam optimizer with an initial variable learning rate of 1e-3, this variable learning rate is controlled by the LR scheduled ReduceLROnPlateau that will be modified if every 5 iterations

Fig. 3. Simple CNN



there are no changes in the validation with a limit up to 1e-8. The RMSE (Root mean square error) and the MAE (mean absolute error) were used as metrics.

A. Loss functions

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2} \quad (9)$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |e_i| \quad (10)$$

B. Training history

5. RESULTS AND DISCUSSION

A. Direct boxels approach + Simple CNN

Dataset	Metric	validation	Test	Inference Time
Direct Boxels	RMSE	1.237	1.193	0.00353
Direct Boxels	MAE	0.953	0.925	0.00353
Direct Boxels	R2	-0.477	-0.403	0.00353

Fig. 4. CNN + XYZ

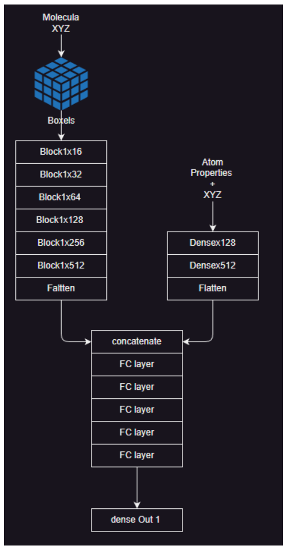


Fig. 5. The R2 during training

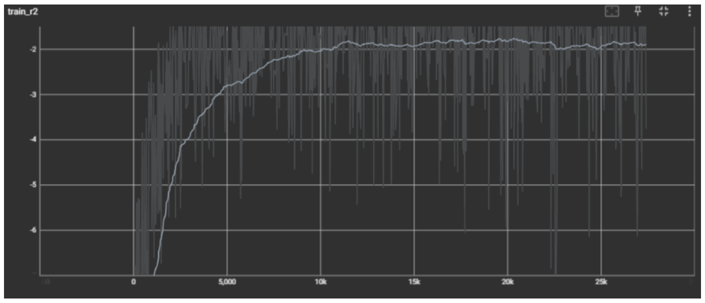
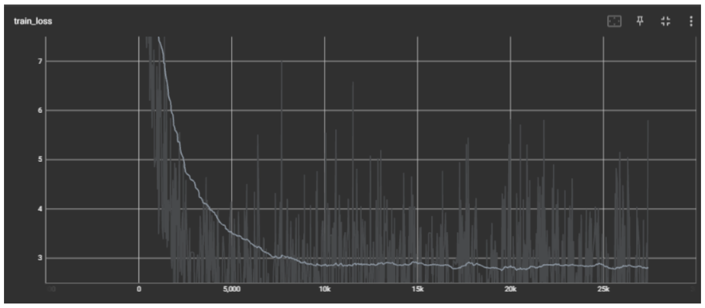


Fig. 6. The RMSE during training


train_loss
10k
15k
20k
25k