Optimized Crystallographic Graph Generation for Material Science

Astrid Klipfel\textsuperscript{1,2,3}, Yaël Frégier\textsuperscript{3}, Adlane Sayede\textsuperscript{2} and Zied Bouraoui\textsuperscript{1}

\textsuperscript{1}Univ. Artois, UMR 8188, Centre de Recherche en Informatique de Lens (CRIL), F-62300 Lens, France. \\
\textsuperscript{2}Univ. Artois, UMR 8181, Unité de Catalyse et de Chimie du Solide (UCCS), F-62300 Lens, France. \\
\textsuperscript{3}Univ. Artois, UR 2462, Laboratoire de Mathématiques de Lens (LML), F-62300 Lens, France. \\
\texttt{\{astrid.klipfel,yael.fregier,adlane.sayede,zied.bouraoui\}@univ-artois.fr}, Xie et al., 2022; Long et al., 2021; Ekström Kelvinius et al., 2022; Gibson et al., 2022]. Notice that organic molecules are composed of wide carbon chains with a limited variety of atoms, while crystal materials are three-dimensional periodic structures composed of a wide variety of chemical bonds and atoms. The periodic structure of crystals is often represented as a parallelepiped tiling, a.k.a. crystal lattice or unit cell. While generating graph-based representations of organic molecules is straightforward, the periodic structure of crystals makes difficult graph processing when training a generative model, and in particular when a massively parallel environment is required. More precisely, generative models may update the geometry of a chemical structure during forward propagation. However, since the graph associated with a given structure is built from the local environment of atoms, a modification of the geometry leads to the modifications of the graph associated with the structure. Consequently, building a generative model with a dynamic graph is hard to achieve on a periodic structure compared to organic molecules.

When training graph-based generative models for material discovery, cutoff distance is a commonly used technique [Schütt et al., 2017; Gasteiger et al., 2020b; Jørgensen et al., 2018]. It designates a relative distance threshold value above which no interaction between nodes is considered. In the same vein, [Jørgensen et al., 2018; Chen et al., 2019] suggests that k-nearest-neighbours (KNN) graphs can also be a good choice for GNN models. KNN-graph is a type of graph where all the nodes are connected to the k-nearest nodes. When processing small molecules, any naive strategy of computing the interatomic distances is feasible, allowing to compute KNN or cutoff graph in a short amount of time and reasonable memory. However, for periodic structures which are infinite, the search area should be carefully selected to avoid unnecessary calculation and memory saturation. In fact, the volume of the search space expands with the cube of the search radius. As such, possible graphs should be generated in milliseconds to be usable in practice during the training process. Moreover, a periodic structure is represented with a multi-graph where a given node can share multiple edges with another and with itself which brings more complexity to the graph generation process. Finally, for big structures, a processing strategy suitable for massively parallel environments should be used in order to deal with a batch of multiple structures at the same time.

Abstract

Graph neural networks are widely used in machine learning applied to chemistry, and in particular for material science discovery. For crystalline materials, however, generating graph-based representation from geometrical information for neural networks is not a trivial task. The periodicity of crystalline needs efficient implementations to be processed in real-time under a massively parallel environment. With the aim of training graph-based generative models of new material discovery, we propose an efficient tool to generate cutoff graphs and k-nearest-neighbours graphs of periodic structures within GPU optimization. We provide pyMatGraph a Pytorch-compatible framework to generate graphs in real-time during the training of neural network architecture. Our tool can update a graph of a structure, making generative models able to update the geometry and process the updated graph during the forward propagation on the GPU side. Our code is publicly available at https://github.com/aclipf/mat-graph.

1 Introduction

New materials discovery is a fundamental challenge in material sciences where high-throughput screening based on machine learning models is largely employed to obtain materials with desired properties. Crystalline (crystal) material generation has recently received considerable attention, e.g. [Xie et al., 2022; Gibson et al., 2022; Klipfel et al., 2023]. In our setting, we are interested in generating new crystall materials for developing new solar panels with a band gap enabling hydrolyse. This helps to solve problems related to clean energy production and storage, which is one of the major challenges facing our society. It can also be used to produce hydrocarbons from CO2, helping to reduce the carbon footprint of human activities.

From organic chemistry to material science, Graph Neural Networks (GNN) have received increasing attention in a variety of tasks such as classification [Schütt et al., 2017; Jørgensen et al., 2018; Gasteiger et al., 2020b; Gasteiger et al., 2020a; Chen et al., 2019; Choudhary and DeCost, 2021; Klicpera et al., 2021] and generation [Satorras et al., 2021; Xie et al., 2022; Long et al., 2021; Ekström Kelvinius et al., 2022; Gibson et al., 2022].
To address the aforementioned issues, we propose an efficient tool that solves KNN and cutoff graph generation for crystalline materials. We provide a compatible implementation with PyTorch that performs on GPUs\(^1\). We used an approach inspired by the KD-tree search algorithm adapted for periodic structures and propose a data structure adapted to massively parallel environments (GPUs) that effectively keeps track of the KNN of each atom. We empirically show the benefits of using our tool.

2 Crystallographic Graph Generation

A crystalline structure can be defined with a cloud of atoms and a repetition pattern that represent periodicity. The repetition pattern is often described as a parallelepiped called a lattice or a cell. The periodic structure is obtained with the tiling of the space by the crystal cell. Consequently, a given atom inside of the cell has multiple positions because of the tiling in space and the local environment of an atom which can overlap with adjacent repetition.

2.1 Crystallographic Graph

We follow [Klipfel et al., 2023] to define the graph associated with a crystal material as an oriented graph where each edge is represented by triplets containing the index of the source node, the index of the destination node and the relative cell coordinate of the destination node. Figure 1 illustrates this representation. Notice that the definition of graph provided in [Klipfel et al., 2023] generalizes to most of the graph definitions proposed in previous works [Jørgensen et al., 2018; Chen et al., 2019; Satorras et al., 2021]. We now recall the formal definition of crystalline structure [Klipfel et al., 2023].

**Definition 1.** The representation space of featured materials \(\mathcal{M}_F^n\) is the disjoint union \(\bigsqcup_{n \in \mathbb{N}} \mathcal{M}_F^n\) where:

\[
\mathcal{M}_F^n = \{(\rho, x, z) \mid \rho \in GL_d(\mathbb{R}), x \in [0, 1[^n \times d, z \in \mathbb{F}_n^3}\}
\]

Chemical materials are represented in \(\mathcal{M} = \mathcal{M}_1^N\), with atomic numbers as feature sequence \(z\).

\(\mathcal{M}_F^n\) is an infinite set of triplet \(\rho, x, z\) that represents all possible materials with \(n\) atoms. The atomic number has a chemistry reference, e.g., \(1\) for hydrogen or \(6\) for carbon.

\(\footnotesize{\text{\textsuperscript{1}Code available at https://github.com/aklif/mat-graph}}\)

2.2 Generation Process

To handle the periodic nature of crystalline, we adapt our graph generation process to work in a torus space. To this end, graph generation is performed by exploring the direct repetition of a cell where we start by evaluating the adjacent cell and extend the search area until we find all the edges. Our graph generation method is built upon two main parts: a searching algorithm and an ordered stack. Combined, the generation process follows an iterative process limiting the RAM usage by splitting the search area. Our generation process remains fast since only a few iterations are required, avoiding useless search areas.

**Searching procedure** Our search procedure is based on a classic KD-tree search strategy. As shown in Figure 2, a search radius is used to represent the area where connected nodes can exist. On the other side, we expand the explored...
area up to a search radius. As the search radius is defined with the KNN in the case of a KNN-graph, the search radius decreases over time when a new area is explored. The search procedure pseudo-code is given by Algorithm 1.

Algorithm 1 KNN graph generation algorithm

Input:
k: the k nearest connected atoms
ρ: the shape of the lattice of the crystal
x: the position of the nodes inside the lattice

Output: a set of edges

1: \(d_i^{\text{max}} \leftarrow \infty\)
2: \(\text{border} \leftarrow (0, 0, 0)\)
3: \(\Gamma_1 \leftarrow \emptyset\)
4: while any\((d_i^{\text{max}} > \text{closest_distance} (\text{border}, \rho))\) do
5: \(\text{extension} \leftarrow \text{next_evaluated_area}(\text{border}, d_i^{\text{max}})\)
6: \(\gamma \leftarrow \text{evaluate_area}(\rho, x, \text{extension})\)
7: \(\Gamma_1 \leftarrow \text{push_nearest}(\Gamma_1, \gamma, k)\)
8: \(\text{border} \leftarrow \text{border} + \text{extension}\)
9: \(d_i^{\text{max}} \leftarrow \max(\{d_i^{\text{max}}, \gamma\})\)
10: end while
11: return \(\Gamma_1\)

Ordered stack To keep track of the k closest points already discovered by our search procedure, we propose an efficient data structure to store points. Our ordered stack first concatenates new data and then sorts them by distance. After that, the edges are filtered to keep only the KNN in the case of a KNN-graph or the edges under a given cutoff distance in the case of a cutoff-graph.

Algorithm 2 Push edges in and ordered stack

Input:
k: the k shortest edges
\(\Gamma_1\): a list of edges
\(\gamma\): the list of edges to merge

Output: the list of the k shortest edges

1: \(\Gamma'_1 \leftarrow \Gamma_1 \parallel \gamma\)
2: \(\Gamma'_1 \leftarrow \text{sort_by_distance}(\Gamma'_1)\)
3: \(\Gamma'_1 \leftarrow \text{stable_sort_by_source_index}(\Gamma'_1)\)
4: \(\delta_i^{\text{max}} \leftarrow \text{k_nearest_distance}(\Gamma'_1)\)
5: \(\Gamma'_1 \leftarrow \{(i, j, \tau) \in \Gamma_1 | d_{(i, j, \tau)} \leq \delta_i^{\text{max}}\}\)
6: return \(\Gamma'_1\)

In addition to graph generation, our tool provides additional functionalities such as:

- Symmetric graph: as some GNN require symmetric directed graphs to perform specific message-passing schema, our tool includes a procedure that makes a given graph symmetric by adding missing edges while guaranteeing the uniqueness of the edges.

- Triplets generation: We provide an implementation to generate triplets composed of two edges sharing the same source nodes during the run-time. This task is important because recent works use triplets information during inference [Klipfel et al., 2023; Xie et al., 2022; Klicpera et al., 2021].

3 Performance Evaluation

To evaluate the performance of our tool, we conducted experiments on Materials project [Jain et al., 2013] which is a dataset composed of 133420 crystalline materials studied with \textit{ab initio} calculation. We considered the same setting as [Xie et al., 2022] where structures composed of more than 64 atoms are removed since they are in general considered outliers. The experiments are performed on an Nvidia quadro RTX 8000 GPU.

CPU vs GPU We compared the time required to process all the structures for our tool with and without GPU optimization. We used a fixed batch size of 256 structures and generated the structures for the 16, the 32 and the 64 nearest neighbours of atoms. As shown in table 1, the KNN-graph generated on GPU is up to 40 times faster than an equivalent CPU library.

Complexity, inference time and RAM usage To check the time complexity of our method, we compare the generation time of one batch with various KNN settings in Table 1 and batch size in Table 2. Experiments on batch size have been performed for a KNN-graph with 32 neighbours.

4 Conclusion

We propose an efficient tool to convert crystalline materials into graphs. Our library allows for reducing the time spent during preprocessing. More importantly, the graph conversion is quick enough to be used during the training process without the preprocessing step and updates the graph while updating the geometry of a given structure. Our tool opens new possibilities in generative networks for material science.
Acknowledgments
This work has been supported by ANR-22-CE23-0002 ERIANA, ANR-20-THIA-0004 and by HPC resources from GENCI-IDRIS (Grant 2022-[AD011013338]).

References


