Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingeniería Agronómica, Alimentaria y de Biosistemas

Computational Biology Master

Application of Graph Neural Networks for the discovery of new materials for the cathode of batteries

Author: Eduardo Abenza Severá

Thesis Director: Roberto Gómez-Espinosa Martín

Institution: HI Iberia, Ingeniería y Proyectos S.L.

Index of Contents

Abstract

The development of batteries with better properties is one of the most promising lines of work in Materials Science nowadays. Because the materials discovery process is slow and expensive, extensive efforts are being made in this line of research. One of these initiatives is LiOn-HD, the project in which this work has been developed. The goal of this project is to discover new materials with higher energy density, lower cost and more environmentally friendly. In LiOn-HD project, the role of HI-Iberia, the institution where this work has been developed, is to use Artificial Intelligence techniques for the discovery of novel materials for their use in the cathode of the batteries. The cathode is usually the component that limits the batteries' performance. Our strategy is to combine a generative model with a predictive model; while the first generates materials, the second one predicts some properties of interest of these new materials that are related to the performance of the material in the cathode. The materials can be represented in different ways for Deep Learning models to learn some property from them. The most promising approach for materials property prediction is to represent the material's unit cell as a graph. Graph Neural Networks are the most adequate Deep Learning models to learn from graphs. Several Graph Neural Networks models have emerged in the last years with an immense predictive power in the field of Materials Science. Two of them, namely Crystal Graph Convolutional Neural Networks and Materials Graph Network, have been evaluated in this work. Another goal of this work was to extract knowledge on why these models work so well. For this objective, XGNN method has been employed for the explanation of Crystal Graph Convolutional Neural Networks predictions in terms of the material's graph structure.

1. Introduction

1.1. Current challenges in Materials Science

We interact with materials daily, as they are an essential part of multiple technological advances. From the screen of our phones to the tires of the cars we drive, their composition results from decades of research in Materials Science. It would be more convenient that this process of design of new materials with specific properties was accelerated. However, nowadays we face some problems in this area: the features of materials are highly dimensional, the chemical search space for the design of new materials is huge and the chemistry and physics knowledge of materials is still insufficient[1].

Apart from the design of new materials, the calculation of their properties is also a challenge. Computational methods such as **Density Functional Theory (DFT)**[2] or **Molecular Dynamics**[3] address this problem. However, their cost in terms of computation and time makes them limited. Also, they are not completely accurate in the values they compute. These issues might be solved with **Artificial Intelligence (AI)** and **Machine Learning (ML)**, because of their strong capabilities in handling massive amounts of high dimensional data (**Figure 1**).

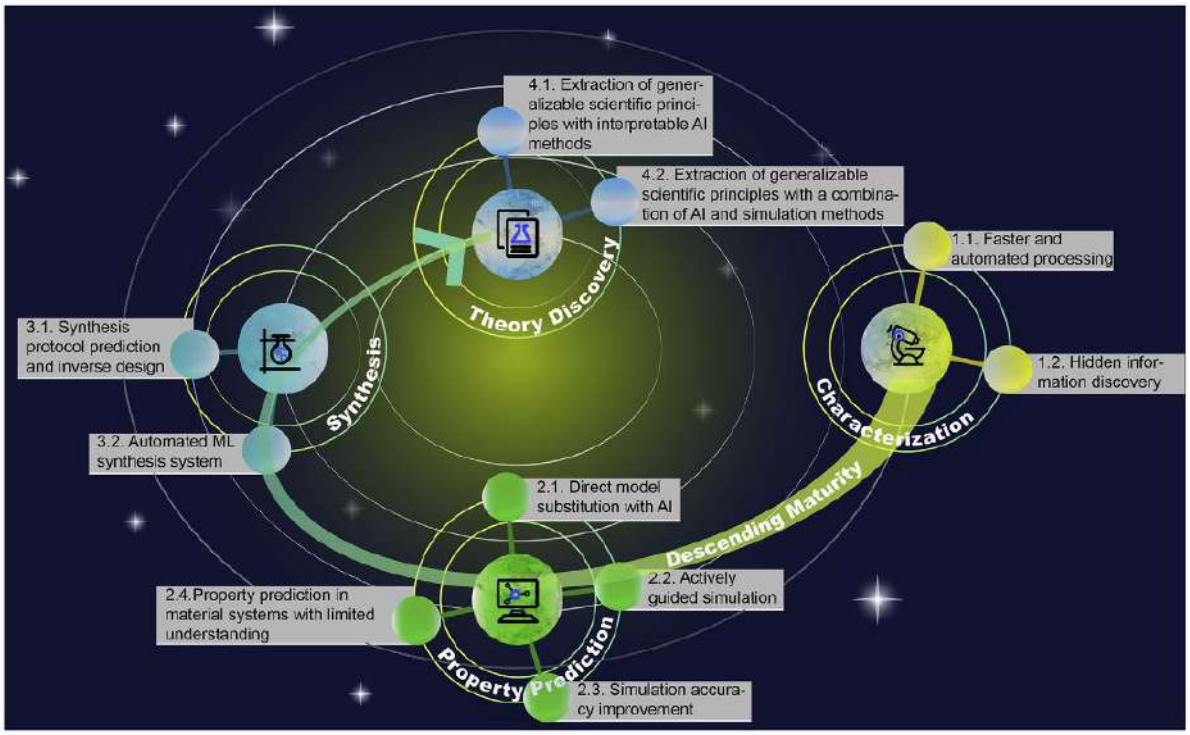


Figure 1. Applications of Artificial Intelligence in Materials Science[1].

Nowadays, there are multiple lines of work in progress related to Materials Sciences. Some of them include COVID-19[4], quantum computing hardware technologies[5] and energy storage[6]. Related to energy storage, this work forms part of **LiOn-HD** project[7], whose goal is to develop new batteries with improved energy efficiency, lower cost, and lower carbon footprint. Specifically, the institution where this work has been developed (**HI Iberia**[8]) is focused on the cathode of the batteries, and the strategy is to develop a framework where a generative model produces new materials and a predictive model (such as Graph Neural Networks explored here) guides training by vaticinating how well those new materials would work in the cathode of batteries. The novel materials that obtain the best results with this model will be tested experimentally in the **Instituto de Ciencias Materiales de Madrid** (an Institute of the 'Consejo Superior de Investigaciones Científicas', CSIC).

1.2. Essentials of Machine Learning

Citing Aurélien Géron, "Machine Learning is the science (and art) of programming computers so they can learn from data"[9]. This tool is useful for solving problems that are either overly complex for traditional approaches or that have no known algorithm. Depending on the task ML methods try to solve, we distinguish three main categories: Supervised, Unsupervised and Reinforcement Learning (**Figure 2**).

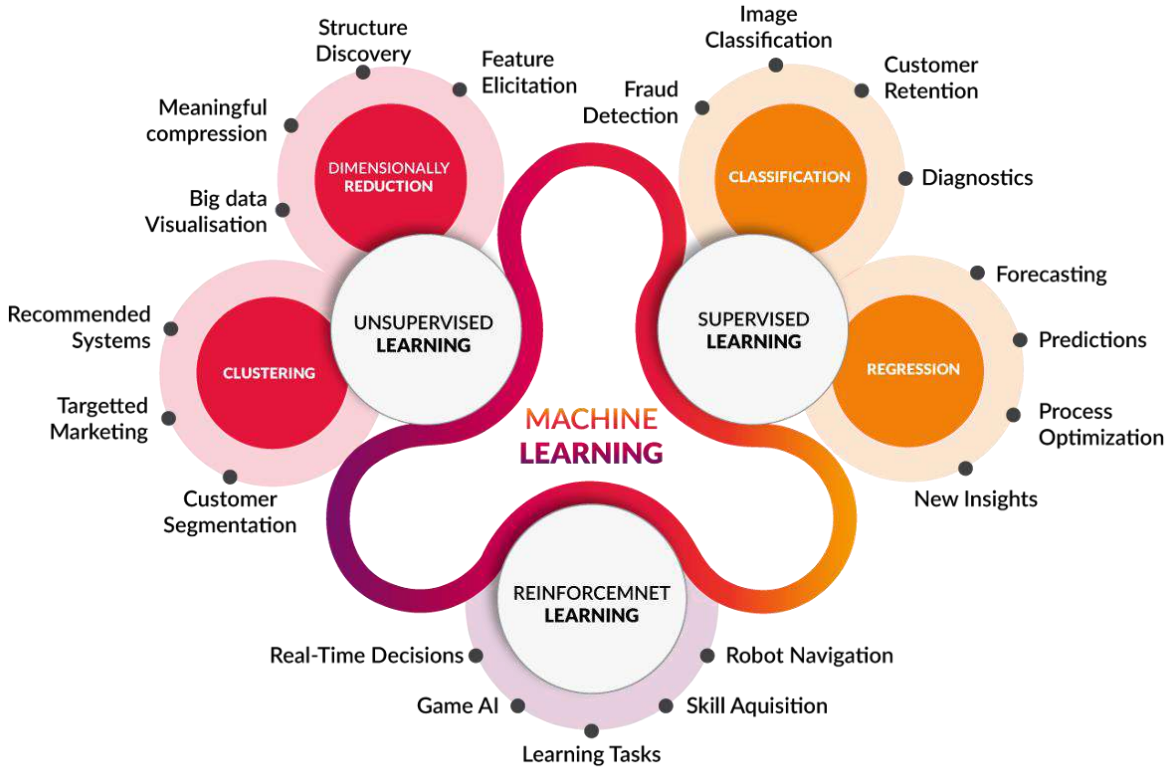


Figure 2. Tasks addressed by Machine Learning in different paradigms[10].

In **Supervised learning** methods we feed the ML model the data associated with the values we want the model to predict (**Figure 3**)**;** depending on the nature of these solution values, we find classification (categorical values) and regression (continuous values) tasks. Some methods in this category include **Support Vector Machines**, **Decision Trees** and **Artificial Neural Networks**. This last method can be adapted to work in any of the other categories and is the core of Deep Learning (DL), a concept that will be reviewed later in this section.

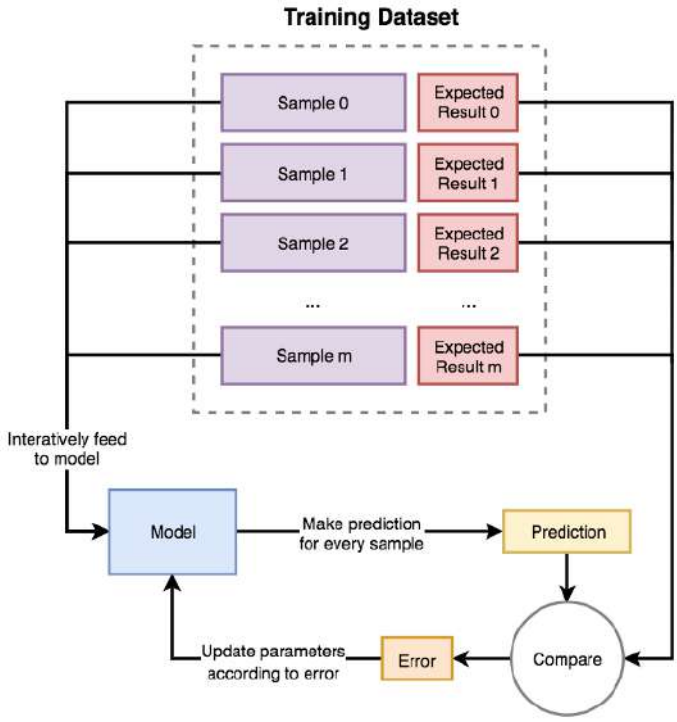


Figure 3. Diagram depicting how Supervised Learning works[11].

Unsupervised learning methods consist of training a ML model with data that is not labelled (**Figure 4**). Some popular unsupervised learning approaches are **K-means** and **Hierarchical Clustering**.

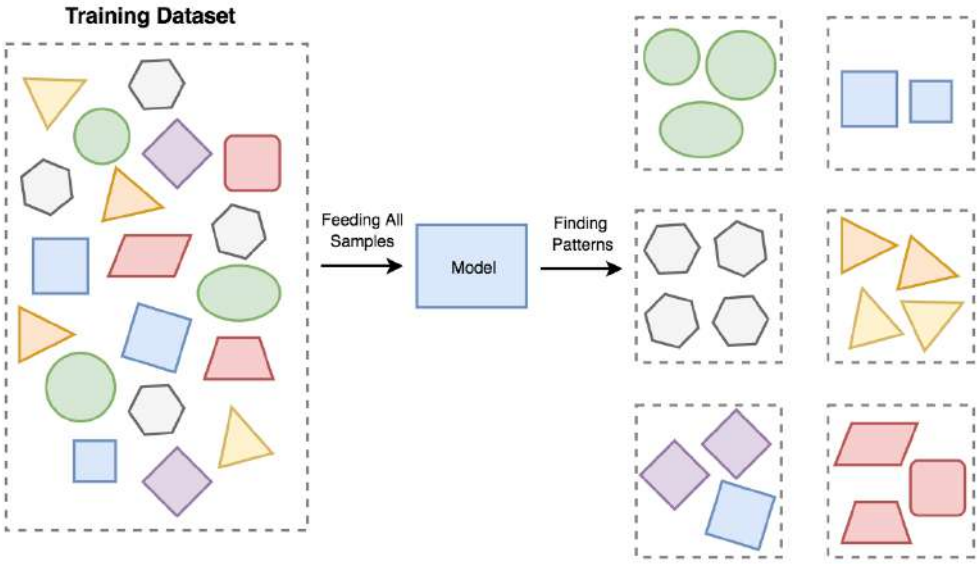


Figure 4. Diagram summarizing how Unsupervised Learning works[11].

In **Reinforcement Learning** methods, there is an agent that sees an environment and, after performing some action that depends on what it has seen, receives a reward or a penalty (**Figure 5**) that helps the agent in choosing better actions in the future. In this context, the agent must learn which is the best policy that it must follow to get the highest reward over time.

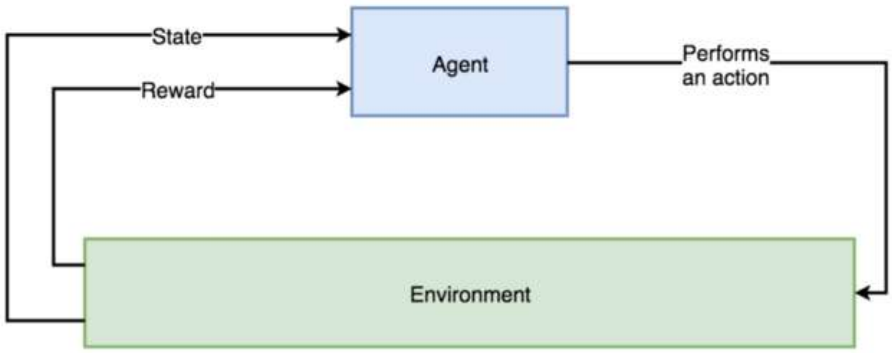


Figure 5. Schema of Reinforcement Learning[11].

Apart from these three main categories we can find more in the literature. For example, there are situations when we have lots of data points and only a small number of them are labelled; **Semi-supervised learning** methods can deal with this kind of data. One popular method that falls within this category is **Restricted Boltzmann Machines**.

In words of Dong et al., "**Deep Learning** is nothing but many classifiers working together, which are based on linear regression followed by some activation functions"[12]. DL is a subset of ML based on neural networks with more than two layers that simulate the behaviour of the human brain, clustering data and making predictions with very high accuracy[13]. This field has expanded considerably in the last 30 years and explaining it in detail is out of the scope of this work. However, advances in Materials Science yielded by DL will be discussed in **section 1.3**.

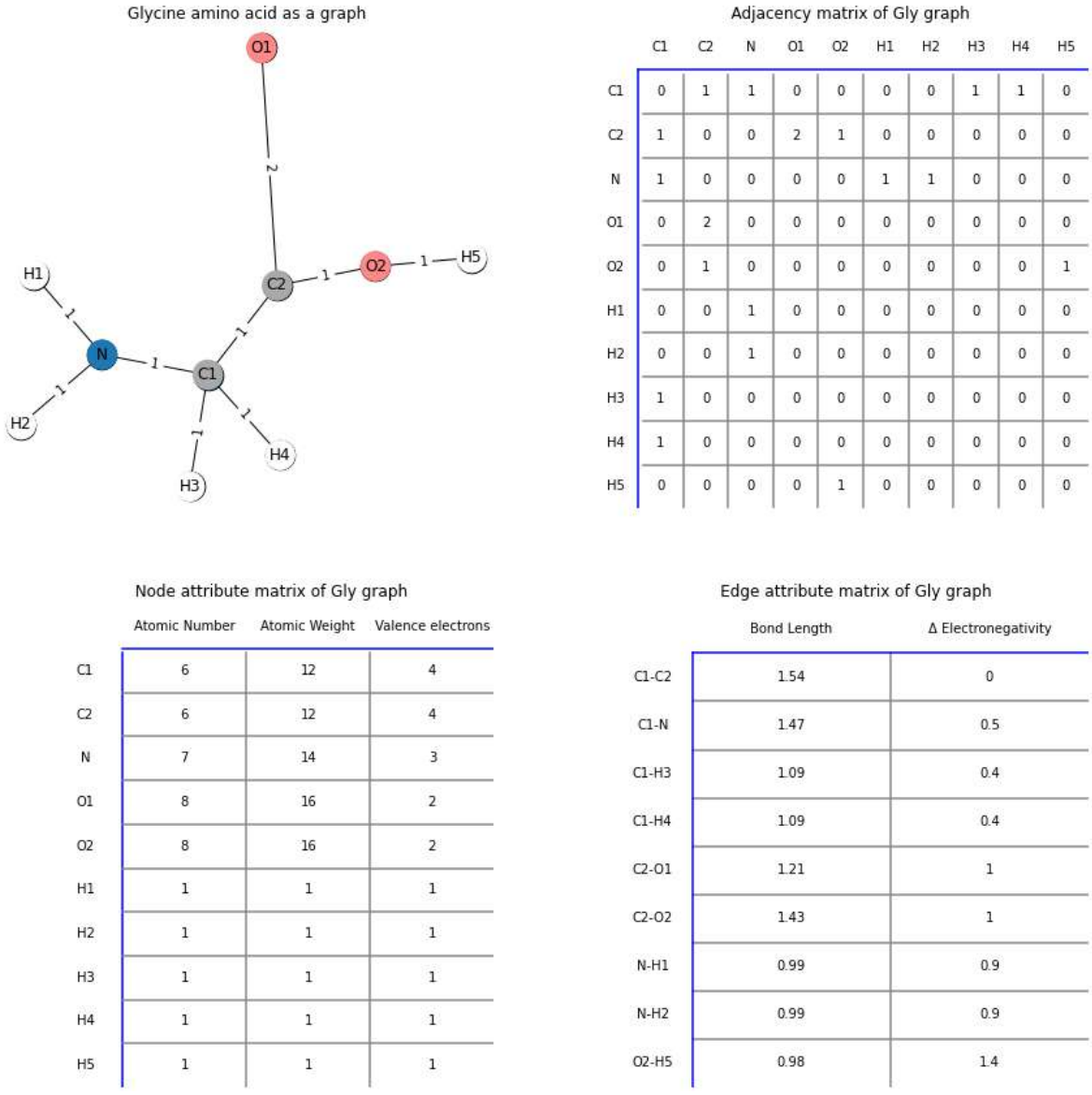
1.3. State of the Art in Artificial Intelligence applied to Materials Science

Some fundamental lines of work in Materials Science are the characterization of materials (calculation and collection of information about a material), material property prediction, material synthesis (determination of one or more possible pathways to synthesize a material of interest) and theory discovery (observation of novel phenomena and extraction of generalizable scientific principles). Examples of how AI has been applied in these areas of Materials Science were widely described by Li et al.[1].

With the application of ML on materials comes the issue of how materials should be represented for ML models to learn from them. **Crystal materials**, which are the scope of this work, are formed by a **unit cell** (a parallelepiped that holds the smallest repeating unit with structure symmetry) that is reproduced along the space's three dimensions. One choice is to transform this unit cell as an **electron density 3D matrix**[14]. Also, some software packages allow the transformation of the atomic structure of the material's unit cell into a **numerical fingerprint**[15]. Matrices and numerical fingerprints can be fed to a traditional DL model (such as a multilayer perceptron or a convolutional neural network) for its training. However, the most promising approach seems to be the representation of the unit cell structure as a **graph**[6].

A graph is a data structure that comprises a set of instances or objects in which some pairs of instances are somehow related. In a graph, the instances are called **nodes** or vertices, and a pair of nodes that are related is connected by an **edge** or link (and the two nodes connected by an edge are **neighbours**). In **Figure 6** (top-left), nodes are represented by circles labelled with letters and edges are represented by lines with associated numerical weights. Attending to the directionality of its edges, a graph can be directed (edges have a direction) or undirected (as in the example, relationships are always reciprocal).

The presence of edges in a graph is summarized in an **adjacency matrix,** where position [i,j] is 1 when nodes i and j are connected by an edge, and 0 elsewhere (binary adjacency matrix); the adjacency matrix is not always binary, as in weighted graphs where each edge is associated to a weight (**Figure 6**, top-right). Apart from adjacency matrix, there are other matrices of interest for the study of graphs. One of them is the **node attribute matrix**: it is the result of stacking the information about all the nodes in a graph (**Figure 6,** bottom-left). Not only nodes but also edges can hold information related to themselves: in this case, this information is summarized in an **edge attribute matrix** (**Figure 6**, bottom-right).



	C1	C2	N	O1	O2	H1	H2	H3	H4	H5
C1	0	1	1	0	0	0	0	1	1	0
C2	1	0	0	2	1	0	0	0	0	0
N	1	0	0	0	0	1	1	0	0	0
O1	0	2	0	0	0	0	0	0	0	0
O2	0	1	0	0	0	0	0	0	0	1
H1	0	0	1	0	0	0	0	0	0	0
H2	0	0	1	0	0	0	0	0	0	0
H3	1	0	0	0	0	0	0	0	0	0
H4	1	0	0	0	0	0	0	0	0	0
H5	0	0	0	0	1	0	0	0	0	0

	Atomic Number	Atomic Weight	Valence electrons
C1	6	12	4
C2	6	12	4
N	7	14	3
O1	8	16	2
O2	8	16	2
H1	1	1	1
H2	1	1	1
H3	1	1	1
H4	1	1	1
H5	1	1	1

	Bond Length	Δ Electronegativity
C1-C2	1.54	0
C1-N	1.47	0.5
C1-H3	1.09	0.4
C1-H4	1.09	0.4
C2-O1	1.21	1
C2-O2	1.43	1
N-H1	0.99	0.9
N-H2	0.99	0.9
O2-H5	0.98	1.4

Figure 6. Introduction to graphs. A. *Weighted undirected graph that represents glycine amino acid. The edge weights indicate bond type (1 = simple, 2 = double). C: carbon, H: hydrogen, N: nitrogen, O: oxygen.* ***B.*** *Adjacency matrix of the graph in* ***A****; when an edge goes from a node i to a node j, position [i,j] of this matrix is 1, and else it is 0.* ***C.*** *Node feature matrix of the graph in* ***A****; for each node there are three attributes: the atomic number, the atomic weight, and the number of electrons in the atom's valence band.* ***D****. Edge attribute matrix of the graph in* ***A****; for each edge there are two attributes: the bond length (in angstroms), and the electronegativity difference between both chemical elements forming the bond. Source: created by the author of this work.*

The matrices described in the last paragraph might be the input for a typical DL model. Based on this representation, the model can learn from the graph and predict a property of interest. However, contrary to inputs for DL models, these matrices have variable dimensions depending on the number of nodes and edges. Another problem is that there is no node ordering in graphs, so it would make no sense to train a model with these inputs. Therefore, **Graph Neural Networks (GNNs)** were proposed[16][17].

GNNs are DL models which have graphs as inputs, contrary to other architectures whose input is a tensor, a matrix, or a vector. Because graphs are **rotation and permutation-invariant** and can represent complex relationships, GNNs have excellent expressive power. GNNs learn through graph convolutions, transforming the input graph into a convenient vector (graph embedding) that describes the graph. Then, predictions are made from this graph embedding.

GNNs learn the representation of a graph in three steps:

1. For each node, the GNN produces a vector representation. This step is called **message passing**. The GNN learns this vector representation (called **node embedding**) by message-passing: the final vector representation is the combination of the information of the corresponding node and its neighbours in a recursive fashion (**Figure 7** depicts a 2-layer graph convolution). This phase is called "message passing" because each node's "message" (information) is passed along the edges to its neighbours. For example, a GNN would learn the node embedding of node C1 (top-left of **Figure 7**) from a hidden vector representation of itself and a hidden vector representation of its neighbours (nodes C2, N, H3 and H4); recursively, each of these representations is learnt from the node attributes of the corresponding node and the node attributes of its neighbours. Because the neighbourhood of atoms C1 and C2 is different, a GNN would learn a different node embedding for each carbon (even though they are the same chemical element).
2. When the GNN has learnt a vector representation for each node in the graph, the second step (**message aggregation**) is to combine these node embeddings. An example might be the computation of their sum or their average. This gives place to a single **graph embedding**, which is a fixed-length vector representation that describes the whole graph.
3. This graph embedding is the input for a feed forward neural network. This network maps the graph embedding vector to the property of interest. This last step is known as **property prediction**.

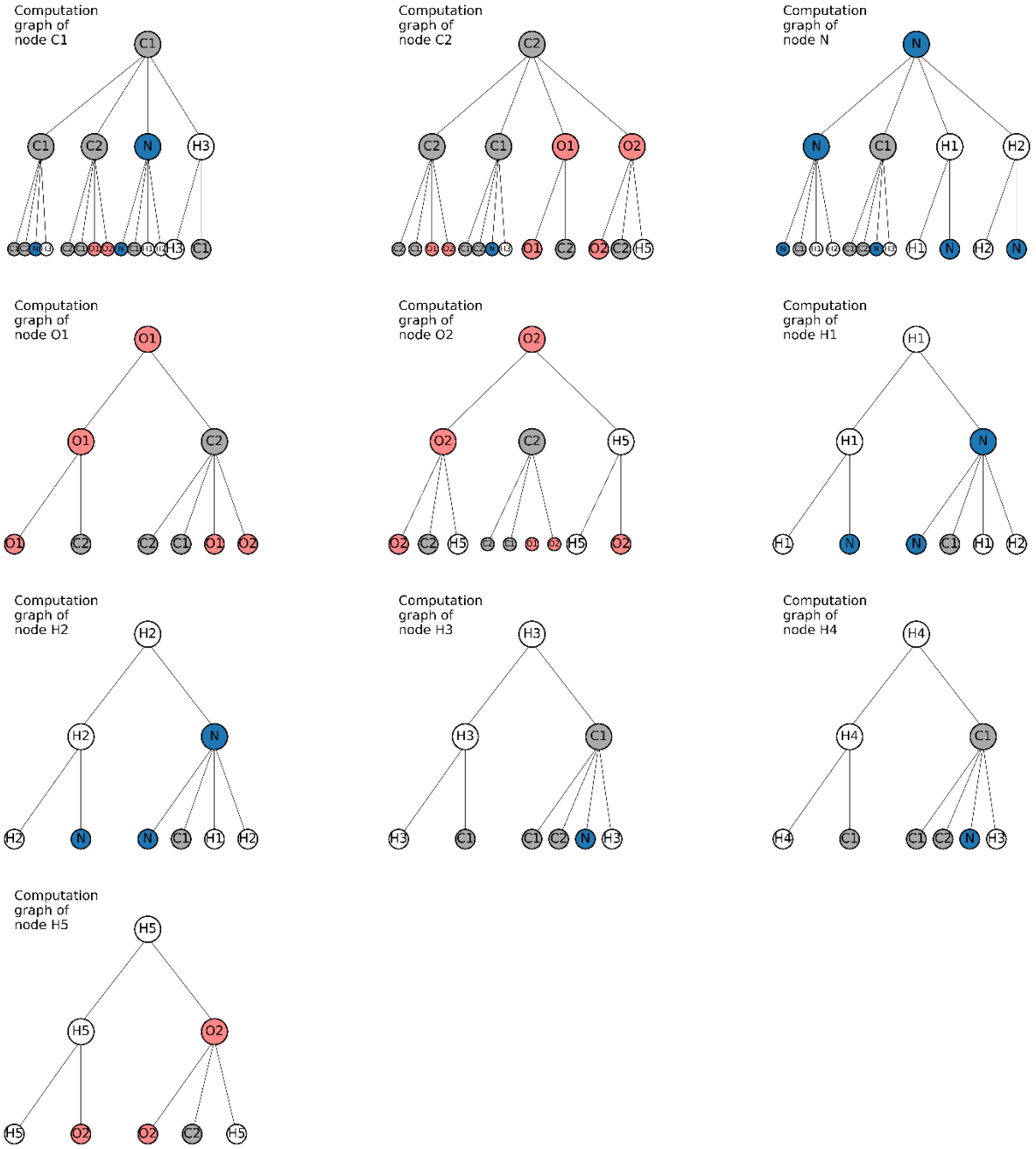


Figure 7. Graphical explanation of message passing phase. *Message passing in all nodes of the graph introduced in* ***Figure 6*** *is shown. This explanation involves a 2-layer graph convolution. For an adequate visualization, only three neighbours per atom are shown (apart from the atom itself); note that in top-left computation graph the node H4 is missing, and so on. In each computation graph we distinguish three stages: bottom (**node attributes** of the nodes), middle (**hidden node representation** obtained after combining the messages in each set of nodes) and top (final **node embedding**). In the message-aggregation phase, the final node embeddings (which are ten vectors in this case) are combined to produce a single vector (**graph embedding**). Source: created by the author of this work.*

Several works have been written using GNNs for the prediction of properties of materials[18][19][20][21][22][23][24]. Also, graphs have been successfully employed as part of models that generate new graphs, even in the field of Materials Sciences[25]. In this work, **CGCNN**[19] and **MEGNet**[23] are used for the prediction of materials properties. For benchmarking, Smooth Overlap of Atomic

Positions (SOAP)[26] descriptors (numerical fingerprints) are fed to two models: Support Vector Regressor[27] and a Multilayer Perceptron.

Because machine learning algorithms usually do not explain their predictions, considerable effort has been made lately to make their decisions more interpretable[28]. Specifically, GNNs have an immense prediction power due to their expressivity, but for us is very hard to explain why they predict a given value for a property of interest. Hence several methods have been developed recently to address this issue[29][30][31][32][33].

In this work we are applying **XGNN**[33] to explain the predictions of GNN models. In a few words, for different target classes (classification tasks), this model finds subgraph patterns in the GNN input graphs with the combination of Reinforcement Learning and hand-made rules. The user of XGNN can specify the number of nodes of the subgraph explanations and some other parameters. In the original work, XGNN was employed for the explanation of predictions on molecules; in this work, this method has been adapted to make explanations on materials.

1.4. Data frameworks of interest in Computational Materials Science

Recently, the production of vast collections of data has augmented considerably. This data generation has been propelled by the computational property prediction of materials with Quantum Mechanics basic principles; methods like Density Functional Theory[2] and Monte Carlo simulations are widely used nowadays.

There are several projects to make available data about materials. Some of them are focused on the production, storage, and organization of **computational data**: The Materials Project[34], AFLOW[35], Open Quantum Database[36], Materials Cloud[37] and NOMAD repository[38]. Other databases store and organize **experimental data**, such as The Pauling File[39], Inorganic Crystal Structure Database (ICSD)[40] and Crystallography Open Database[41]. Some of these resources and others are summarized in **Table 1** (next page). Currently, in the LiOn-HD project, Materials Project, Open Quantum Database and NOMAD repository are being employed and will be introduced in the following paragraphs.

The **Materials Project** (MP)[43] contains theory-based data, web analysis tools and software for the performance and analysis of calculations. This database holds properties of more than 140000 inorganic materials (as of September 2021), and its programmatic use is quite simple with functionalities such as Pymatgen[44] and the Materials API[45].

NOMAD (NOvels MAterials Discovery) is an open framework for Materials Science data gathering, sharing and curation. Most of these data have been calculated computationally with state-of-the-art software packages. One of the main motivations of NOMAD is to generate a massive volume of data to boost the discovery of new materials and of information associated with them. Also, there is an interface called NOMAD API that allows access to the NOMAD database programmatically.

Open Quantum Materials Database (**OQMD**) is a database with DFT-calculated thermodynamic and structural properties for more than 800000 materials. OQMD contains predictions for ICSD structures and hypothetical structures derived from real ICSD structures. There is a Python package called *qmpy* with several tools for computational Materials Science; one of them is *QMPYRester*, which allows queries on the OQMD database.

Table 1. Services provided by some significant Materials Science data infrastructures[42].

	Open Access	Comp. data	Exp. data	Data upload (DOIs)	Workflow management tools	Web API	Data analysis tools
AFLOW	✓	✓			✓	✓	✓
Computational Materials Repository	✓	✓			✓		✓
Crystallography Open Database	✓	✓	✓	✓			
HTEM	✓		✓	✓		✓	✓
Khazana	✓	✓	✓				✓
MARVEL NCCR	✓	✓		✓	✓		✓
Materials Data Facility (MDF)	✓	✓	✓	✓(DOI)*		✓	
Materials Project	✓	✓			✓	✓	✓
MatNavi/NIMS	✓	✓	✓				✓
NOMAD CoE	✓	✓		✓(DOI)		✓	✓
Organic Materials Database	✓	✓					✓
Open Quantum Materials Database	✓	✓					✓
Open Materials Database	✓	✓		✓	✓	✓	✓
SUNCAT	✓	✓				✓	✓
Citrine Informatics	✓**	✓	✓	✓		✓	✓
Exabyte.io						✓	✓
Granta Design		✓	✓				✓
Materials Design		✓	✓				✓
Materials Platform for Data Science	✓***	✓	✓			✓	✓
MaterialsZone			✓				✓
SpringerMaterials			✓				✓

1.5. Applications of Graph Neural Networks in Computational Biology

In this section, some works using **GNNs** for their application in **Computational Biology** will be reviewed. The motivation of this brief review is to highlight the expressive power of GNNs and how they can be applied to solve different problems in Computational Biology.

In 2018, You et al.[46] developed a DL framework that generates new molecules with optimized properties of interest with the combination of Reinforcement Learning, Generative Adversarial Networks and molecular graph representations. In this model, an agent learns a **Reinforcement Learning policy** (called GCPN) when exposed to a graph **generation environment**, where it receives positive or negative rewards based on some **properties of interest and** the **similarity** of the generated molecules with the molecules in a dataset. The environment checks the **validity** of the newly generated molecules considering the valences of the chemical elements in the molecule. GCPN predicts the atoms and bonds to be added to the molecule using a special kind of GNN (**relational GCN**) to learn a representation for distinct types of bonds (simple, double and triple bonds). Some properties optimized with this method were logP and a quantitative estimate of drug-likeliness.

In 2020, Stokes et al.[47] employed a GNN model (**ChemProp**) to **repurpose** already-approved drugs for their use as **antibiotics**. In this work, one of the molecules with the best predicted antimicrobial activity (previously used for the treatment of diabetes, rebranded by the authors as halicin) is tested experimentally, showing its efficacy against a broad spectrum of bacteria and in mice. Also, the

mechanism of action of this drug was studied, and the authors discovered that it acts through the deregulation of the membrane electrochemical gradient of bacteria.

Finally, this year a significant breakthrough in Computational Structural Biology occurred: Jumper et al. published **Alphafold2**[48], which accurately predicts the **three-dimensional structure of a protein given its sequence of amino acids**. The architecture of this model is summarized in **Figure 8A**. This successful method takes an amino acid sequence as input, performs a multiple sequence alignment to detect conserved or novel amino acids in the sequence (MSA) and looks for known structure templates for similar sequences (Templates). Then, the Evoformer learns a representation for MSA (MSA representation) and Templates (Pair representation), and the output of Evoformer is fed to Structure module. A schema of Structure module is shown in **Figure 8B**. Here, authors see the folded protein as a **'spatial graph'**, where nodes are amino acids of the protein and amino acids in proximity are connected by edges. Using an **attention-based** neural network architecture (8 blocks with shared parameters), the structure of this graph is interpreted and reasoned over (with Pair representation and a linear projection of the first row in MSA representation) before it outputs the 3D coordinates of the whole protein[49].

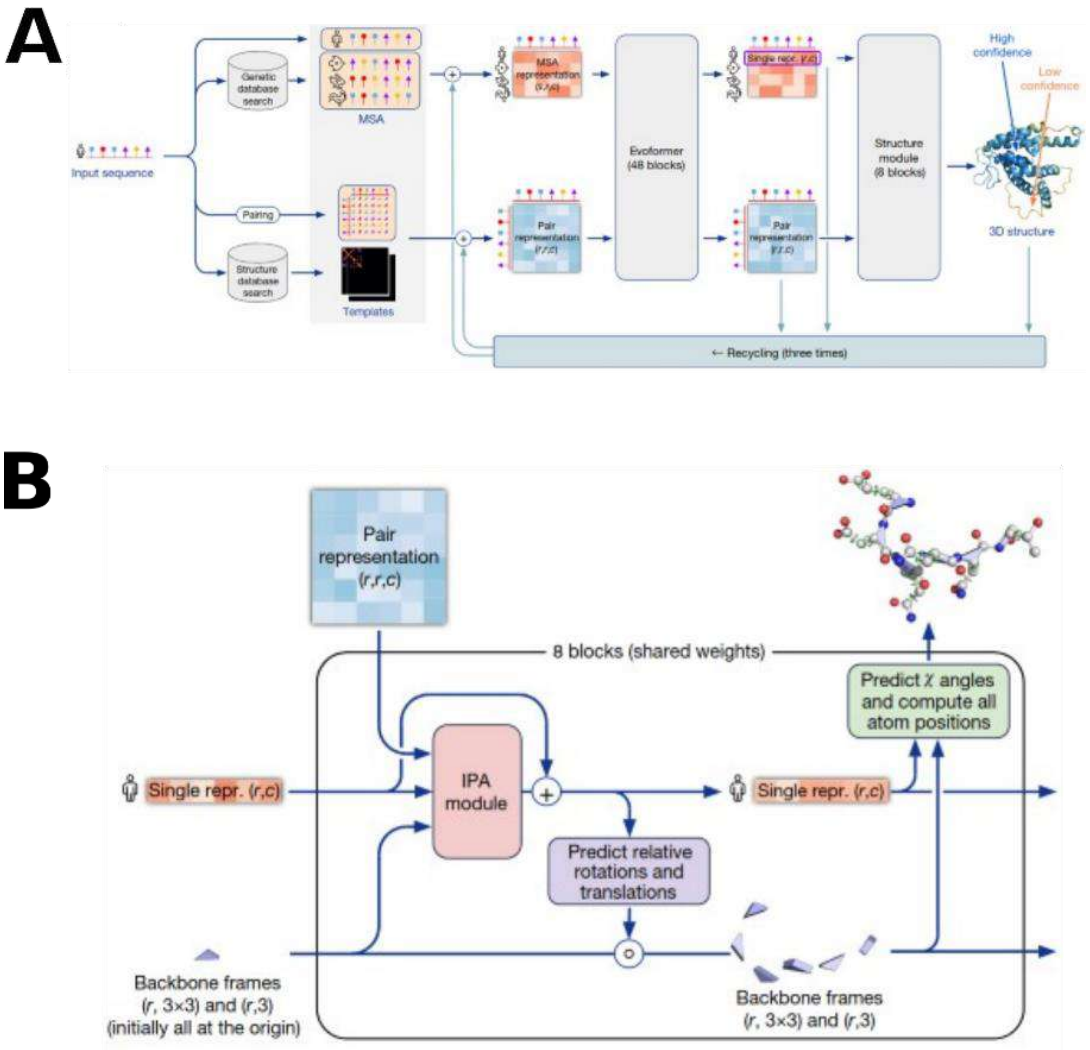


Figure 8. AlphaFold2[48]. A. *Architecture of AlphaFold2 model.* ***B****. Architecture of Structure module. For the prediction of 3D coordinates, the protein is represented as a graph.*

1.6. Objectives

The general objective of this work is:

- Study and analysis of different Graph Neural Networks models and their application on the discovery of new materials for the cathode of batteries.

The specific objectives of this work are:

- Comparison of the predictive power achieved by several Graph Neural Networks models with traditional methods (ML and DFT).
- Comparison of the computational cost of these methods.
- Interpretation of the problem of graph generation from the 3D structure of materials.
- Explanation of the predictions of a Graph Neural Network.

2. Materials and Methods

2.1. Databases

In this work we have developed a data collection framework that obtains the desired properties from **Materials Project**, **OQMD** and **NOMAD** databases. The properties have been standardized where possible across databases (units and datatypes). The datasets employed in this work are based only on the Materials Project; hence databases NOMAD and OQMD will not be explored in the rest of this work.

2.1.1. Materials Project

The **Materials Project** is a database that offers theory-based data, in addition to web-based functionalities for the analysis of materials, and software for several uses related to Materials Science (such as **Pymatgen**[44]). This database holds information about more than 140000 different inorganic materials, which are the scope of this work.

We have used Materials API's[45] MPRester class (`pymatgen.ext.matproj.MPRester`) to retrieve information about the materials, using MongoDB operators within its query method. Some material's properties obtained from this database are:

- **Structure-related**: the CIF file of the material, the cartesian coordinates of the atoms in the unit cell, the formula of the material or the space group symbol it belongs to.
- **Lattice-related**: angles, lengths and matrix that define the lattice of the material.
- **Others**: energy of the material, formation energy per atom, the band gap or the ICSD ID.

This knowledge has been applied to generate the datasets described in **section 3.1**.

2.2. Software libraries

For this work, Python 3 has been used in an Ubuntu 20.04 computer. To help with code writing and execution, several libraries have been employed:

- **Numpy**[50]: fast computing.
- **Pandas**[51]: for data analysis and manipulation.
- **Matplotlib**[52]: visualization library.
- **Pymatgen**[44]: a package developed by The Materials Project for dealing with materials data. Some essential tools for this work include REST-API for downloading Materials Project data, CIF file parsing and finding the nearest neighbours of each atom in a material.

- **NetworkX**[53]: a library developed for the analysis and visualization of graphs and networks. It has been employed to explore the graphs generated from the material's structures.
- **PyTorch**[54]: Deep Learning framework that combines usability and speed. It is widely adopted by the software development community. This library has multiple machine learning architectures that have been useful for this work.
- **DScribe**[15]: transforms atomic structures into fixed-size numerical fingerprints (descriptors).
- **Scikit-learn**[55]: Python module for traditional Machine Learning.

2.3. Trained models

2.3.1. CGCNN

Crystal Graph Convolutional Neural Networks (**CGCNN**)[19] is a GNN that represents materials by a crystal graph (**Figure 9A**). In this crystal graph the nodes are atoms and, for each atom, edges are added between that atom and a maximum number of neighbours (hyperparameter of the model) closer than a given cut-off distance (another hyperparameter). For each atom in the unit cell of the material, a vector representation is learnt by exploring its neighbourhood (graph convolutions), and all these atom embeddings are concatenated (with mean pooling) in a graph embedding that is used for prediction (**Figure 9B**).

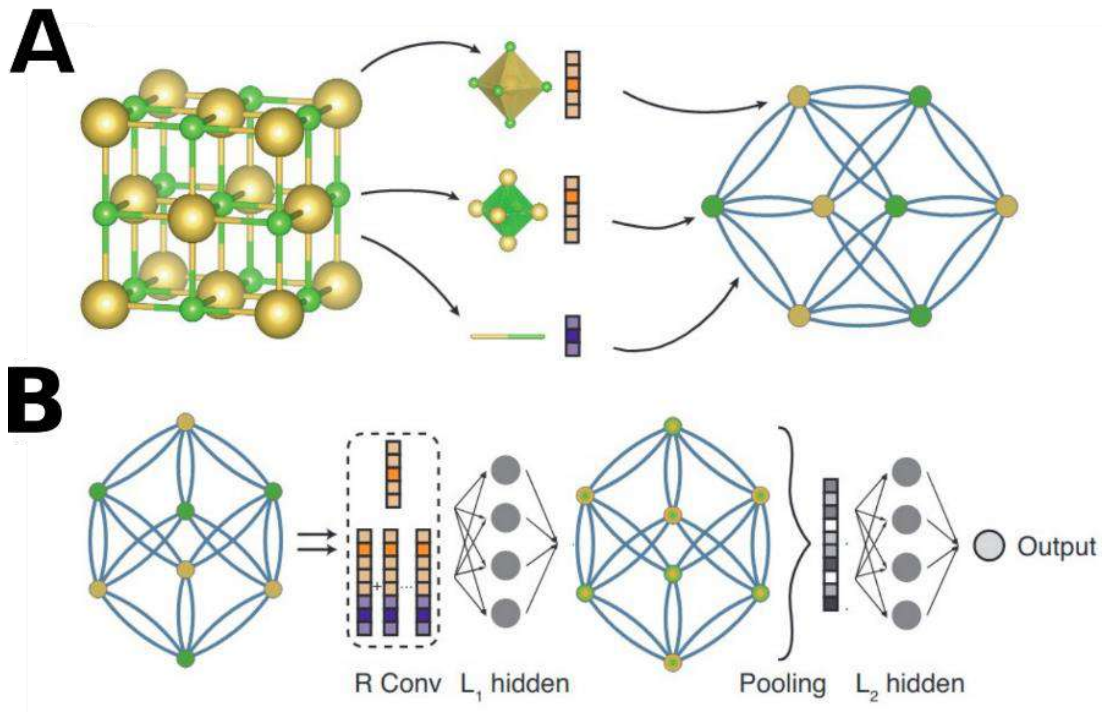


Figure 9. The architecture of CGCNN model[19]. A. *3D structures of materials' unit cell are transformed into graphs.* ***B.*** *Schema of CGCNN model layers. With graph convolutions ("R conv"), the input graph node attributes are updated with the information of each node's neighbours. The node embeddings are converted to a single graph embedding with pooling. The graph embedding is used for making predictions ("L_2 hidden").*

2.3.2. MEGNet

Another model based on graphs that is being evaluated in this work is **MatErials Graph Networks** (**MEGNet**)[23]. In the words of its authors, this model should not be classified as a GNN but as a Graph Network, where it is not compulsory the use of neural networks as function approximators. It can be used to predict the properties of both molecules and crystals.

The input of this model is a graph, where atoms closer than a given cut-off distance are linked by edges (there is not a maximum number of neighbours per atom). In this graph, there are **atom attributes** (such as atomic numbers), **bond attributes** (distance between atoms) and **state attributes** (temperature of the system). The architecture of this model is shown in **Figure 10**.

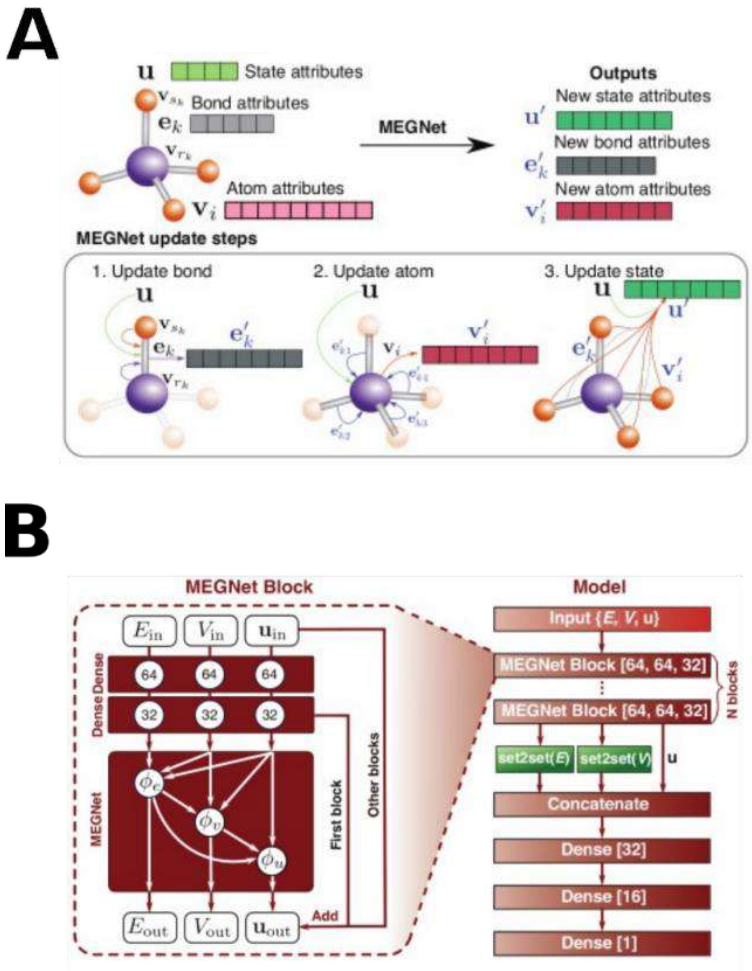


Figure 10. Schema of MEGNet model[23]. A. *MEGNet module, an essential component in the MEGNet model (top). Edge, node and state attributes are updated sequentially with graph convolutions.* ***B.*** *Overview of MEGNet model (bottom). Node embeddings are concatenated with the output of node and edge set2set layers, and with the resulting graph embedding MEGNet model makes predictions (three dense layers).*

2.3.3. SOAP-SVR and SOAP-MLP

There are multiple descriptors for materials apart from graphs. One of them is **Smooth Overlap of Atomic Positions (SOAP)**, which is a numerical fingerprint that "encodes regions of atomic geometries by using a local expansion of a Gaussian smeared atomic density with orthonormal functions based on spherical harmonics and radial basis functions"[56]. Using the DScribe library, SOAP descriptors for the materials were generated with the following parameters:

- Cut-off radius for local region: 5 angstroms.
- Number of radial basis functions: 1.
- Maximum degree of spherical harmonics: 2.
- Averaging over sites: before summing up the magnetic quantum numbers.

The descriptors produced in the previous conditions were fed to two different ML models:

- **Support Vector Regression**. The kernel type used in the algorithm was the Radial Basis Function kernel. Default hyperparameters of **scikit-learn**'s `sklearn.svm.SVR` class were used.
- **Multilayer Perceptron**. The idea here was to directly compare the graph embeddings learnt by CGCNN (**2.3.1**) with SOAP descriptors. SOAP descriptors were fed to CGCNN after pooling layer ("L_2 hidden" in **Figure 9B**).

2.4. Metrics for evaluation of models

The models were evaluated over randomized training:validation:test (80:10:10) splits for 10 (CGCNN and MEGNet) and 2 (SOAP-SVR and SOAP-MLP) iterations. SOAP-SVR does not need validation set, so the dataset for this model's training was split 80:20. The models have been evaluated in terms of:

- **Regression metrics**: because the scope of this work is to predict the properties of the materials quantitatively, only regression is performed. The metrics considered in this work have been MAE, RMSE and R^2.
  - **MAE**: mean absolute error between real and predicted properties.
  - **RMSE**: root mean square error between real and predicted properties.
  - **R^2**: coefficient of determination between real and predicted properties. It is the proportion of variance in the real property values of test set instances that can be explained by the model predictions on test set instances.
- **Computational cost**: not only must models have predictive power, but also, they should supply interesting metrics related to the model's use and deployment. The metrics considered here are model **training time** and model **parameters size**.

2.5. Explainability of the GNN predictions

GNNs have huge predictive power. We know they predict accurately because they learn a convenient vector representation of graphs based on the neighbourhood of nodes and on node, edge and graph attributes. However, it is hard to explain why they predict a property value for a given graph. In this work the method **XGNN**[33] is used for the **explanation of GNN predictions**.

XGNN allows the interpretation of GNNs at the model level after training a Reinforcement Learning graph generator. This method explains the predictions of a GNN model in terms of the **connectivity** between atoms and the **identity** of these atoms.

The graph generator produces graph patterns (subgraphs) that try to maximize a given model prediction matching some validity rules. In this work, the source code of this method has been adapted to CGCNN with two different approaches (atom identity):

- Atoms are labelled with their **periodic group**.
- Atoms are labelled with their **period number**.

The maximum number of atoms per explanation has been set to 6 and the maximum number of neighbours per atom has been set to 4. The properties predicted in the CGCNN paper have been

discretized with the median of each property in the training set: instances lower or equal than the **median** are class "low" and instances above the median are class "high". These categorical properties have been used as a target for classification, and XGNN method has been employed to explain the CGCNN predictions on the test set. Only predictions with a probability higher than 0.8 were allowed. For each property, 300 explanations have been produced (150 per class).

3. Results

3.1. Exploratory analysis of databases

In this work, two datasets have been employed:

- **CGCNN**: this dataset was described in CGCNN paper[19]. In its GitHub repository a list of 46744 Materials Project IDs of entries used in the paper can be found[57]. For every entry, the properties indicated in **Table 2** were retrieved.
- **MEGNet**: this dataset was described in MEGNet paper[23] and is stored in Figshare[58] repository. This dataset contains 69239 materials with 7 properties (described in **Table 2**).

Table 2. Properties in the datasets employed in this work. *A brief description is provided. The presence of the properties in each dataset (CGCNN and MEGNet) is marked with an "X".*

Property	Description	CGCNN	MEGNet
Materials Project ID	An identifier that associates a material to its corresponding entry in the Materials Project.		**X**
Primitive CIF file	A file that contains crystallographic information (structure) of a material's asymmetric unit, i.e., the minimum region of a material's structure that, after the application of the corresponding symmetry and translation operations, forms the entire material's structure.	**X**	**X**
MEGNet graph	Graph generated by MEGNet model from the structure of the material that appears in the 'Primitive CIF file'. Different from CGCNN, this graph is built only considering the distances between atoms (there is no maximum number of atom neighbours).		**X**
Energy	The energy of the Materials Project entry. It is usually the final calculated energy from VASP or other software. It is normalized by the number of atoms in the material.	**X**	
Formation energy	The formation energy of the material divided by its number of atoms.	**X**	**X**
Fermi energy	The electrochemical potential of the material at 0K temperature [59]. In ordinary metals, this value is very close to electrochemical potential under 1000K.	**X**	
Band-gap energy	Distance between the outer part of the valence band and the lower part of the conduction band. Represents the minimum energy that is necessary for exciting an electron to the conduction band (inversely proportional to conductivity).	**X**	**X**
Bulk modulus	Compressibility, in logarithmic scale. Measures resistance to compression of a material.	**X**	**X**
Shear modulus	Modulus of rigidity of a material, in logarithmic scale.	**X**	**X**
Poisson ratio	A measure of the Poisson effect, that is the deformation of a material in directions perpendicular to the specific direction of loading. For example, because an elastic band experiences contraction perpendicular to the direction of stretching, its Poisson ratio is positive.	**X**	

Before performing any analysis in these two datasets, it is advisable to look at the values we are trying to predict. The distributions of the target properties in CGCNN and MEGNet datasets are explored in

Figures 11 and **12**, respectively. I find convenient pointing out that "Formation energy" follows a bimodal distribution in the two datasets, and that there is a considerable number of instances with "Band-gap energy" close to zero; in MEGNet dataset this peak is less pronounced because the instances with this property equal to zero are filtered out. In CGCNN dataset, Poisson ratios lower than 0 and higher than 0.5 are filtered out too.

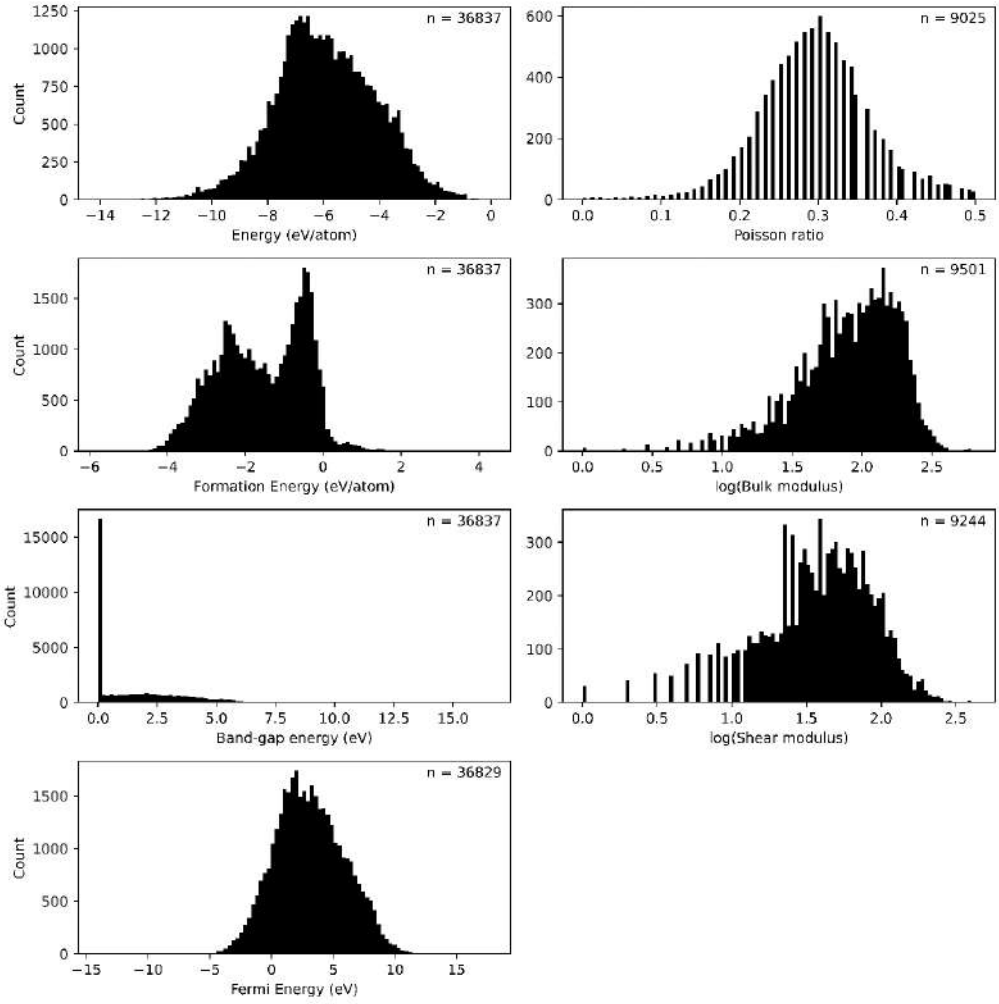


Figure 11. Distribution of values for target properties in CGCNN dataset. *In the upper-right corner of each subplot is highlighted the number of instances (n) with a value for the corresponding property in the dataset.*

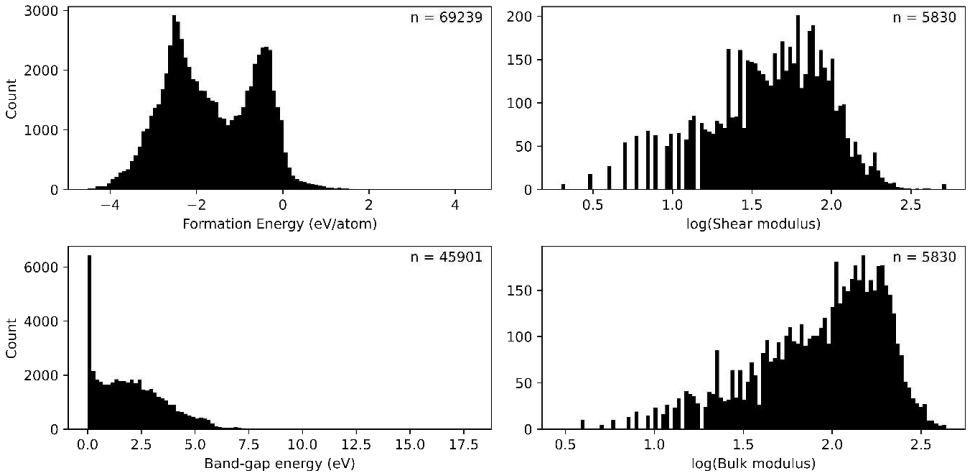


Figure 12. Distribution of values for target properties in MEGNet dataset. *In the upper-right corner of each subplot is highlighted the number of instances (n) with the corresponding property in the dataset.*

3.2. Generation of material graphs

The inputs for GNNs are graphs. Molecules (**section 1.5**) are easily converted to graphs: atoms are represented by nodes and bonds between atoms are edges in the graph. However, this work focuses on crystal materials, and the transformation of these entities into graphs is not that simple.

In CGCNN[19] and MEGNet[23], the strategy for transforming material's structures to graphs is the following:

- Same as with molecules, nodes are atoms in the material.
- A pair of atoms is linked by an edge if the **distance between the two atoms** is lower than a threshold (hyperparameter).

These two methods differ in whether there is a **maximum number of neighbours** allowed per atom. In CGCNN this number is also a hyperparameter, and in MEGNet there is not a maximum number of neighbours per atom. A graphic example of the CGCNN procedure is shown in **Figure 13**.

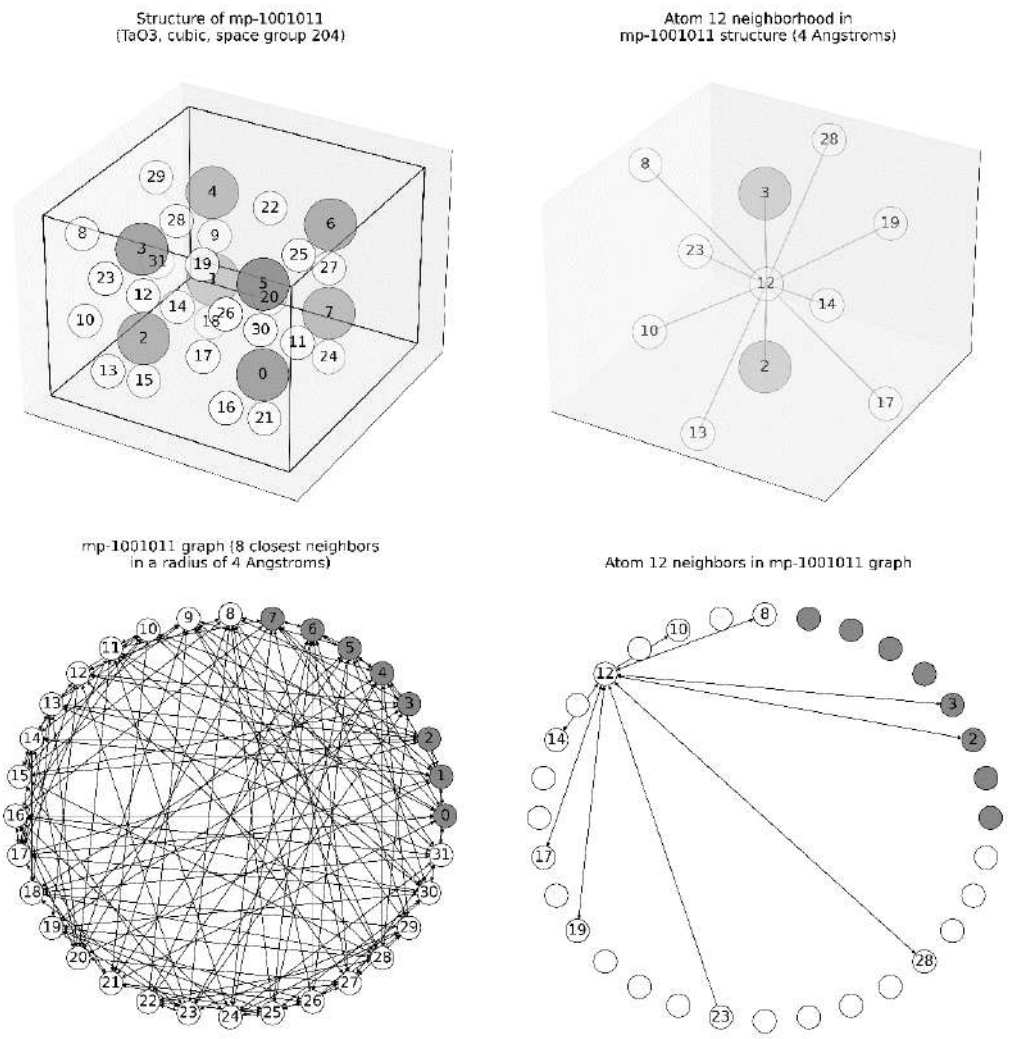


Figure 13. Visual explanation of the graph generation process from material's structures. *Grey: Ta; white: O.* ***A)*** *Atoms in 3D, as specified in CIF file.* ***B)*** *Atoms in 3D, with zoom in atom number 12 proximity.* ***C)*** *Graph generated from material's structure: 8 closest neighbours in a radius of 3 angstroms.* ***D)*** *Graph from* ***C****, showing only the edges between atom 12 (same as in* ***B****) and its neighbours in the graph.*

These two parameters, the maximum number of neighbours and the maximum distance between neighbours, determine the **expressiveness of the GNN model**. A number too low of atom neighbors will result in a graph with very few edges where atoms are isolated, and a number too high would imply that all atoms in the material are neighbours and practically the same vector representation would be learnt for all of them.

3.3. Performance of GNNs versus traditional models

In previous sections, CGCNN and MEGNet datasets were explored and the method for the transformation of materials' 3D structures into graphs was explained. The objective here is to demonstrate how powerful are CGCNN and MEGNet GNNs models for the prediction of several properties of the materials.

The first step was the replication of Xie et al. work[19]. Using **CGCNN** dataset (**section 3.1**), CGCNN model was trained with the best hyperparameters indicated in the original work. The list of hyperparameters and their default values can be seen in **Table 3**. The results obtained in this work, and those obtained by the authors, are shown in **Table 4.**

Table 3. Hyperparameters of CGCNN model. *Default values (best performing hyperparameters in the original work) are shown.*

Hyperparameter	Default value	Hyperparameter	Default value
Epochs	30	Number of graph convolutions	3
Batch size	256	Number of layers after convolution	1
Learning rate	0.01	Maximum number of neighbours per atom	8
Learning rate milestones	100	Cut-off radius for finding neighbours	8
Momentum	0.9	Minimum distance for Gaussian basis	0
Weight decay	0	Step for Gaussian basis distance	0.2
Optimizer	Stochastic Gradient Descent		
Number of hidden atom features in convolution layers	64		
Number of hidden atom features after pooling	128		

Table 4. Results obtained with CGCNN model in CGCNN dataset. *Results obtained in the original work[19] ('paper') and in this work ('work'). Also, DFT results (computational) reported in the original work[19] are included.*

Target property	Units	Training instances$_{paper}$	MAE$_{paper}$	Training instances$_{work}$	MAE$_{work}$	MAE$_{DFT}$
Fermi energy	eV	28046	0.363	36 837	0.473	
Energy	eV/atom	28046	0.072	36 837	0.132	
Formation energy	eV/atom	28046	0.039	36 837	0.084	0.081−0.136
Band-gap energy	eV	16458	0.388	36 837	0.363	0.6
Bulk modulus	log(GPa)	2041	0.054	9 507	0.085	0.050
Shear modulus	log(GPa)	2041	0.087	9 258	0.142	0.069
Poisson ratio	–	2041	0.030	9 024	0.037	

Next, the work of Chen et al. [23] was also replicated. **MEGNet** dataset (**section 3.1**) was employed. For each property, the authors trained MEGNet model for more than 1000 epochs. However, results of models trained for a high number of epochs in this work did not outperform those trained for 30 epochs. Hence, to save time and computational resources, MEGNet models here were trained for only 30 epochs. There hyperparameters were set to default in MEGNet model. The results obtained with MEGNet in this work, and those obtained by the authors, are shown in **Table 5**.

Table 5. Results obtained with MEGNet model in MEGNet dataset. *Results obtained in the original work[23] ('paper') and in this work ('work').*

Target property	Units	Training instances$_{paper}$	MAE$_{paper}$	Epochs$_{paper}$	Training instances$_{work}$	MAE$_{work}$	Epochs$_{work}$
Formation energy	eV/atom	60 000	0.028	>1000	55 391	0.122	30
Band-gap energy	eV	36 720	0.330	>1000	55 391	0.552	30
Bulk modulus	log(GPa)	4 664	0.050	>1000	4 664	0.117	30
Shear modulus	log(GPa)	4 664	0.079	>1000	4 664	0.125	30

Once the results of **CGCNN and MEGNet** were replicated, both models were trained and evaluated on **CGCNN dataset** for a fair comparison between them. Also, in an attempt to compare GNNs (and graphs as descriptors) with other traditional ML methods, two extra models were tested:

- **Support Vector Regression** with **SOAP descriptors** of the materials as input.
- **Multilayer Perceptron** with **SOAP descriptors** of the materials as input.

The results for the four models are summarized in **Table 6** and **Figure 14**.

Table 6. Performance and computational cost metrics of the models under study in this work in the prediction of seven materials properties. *In all cases, CGCNN dataset was employed. Values correspond to mean ± standard deviation. For CGCNN and MEGNet, n=10 (8 with GPU and 2 without GPU for CGCNN). For SOAP-SVR and SOAP-MLP, n=2. The best result per category for each property is highlighted.*

Target property	Model	MAE	RMSE	R^2	Training time (s)
Energy	CGCNN	**0.135±0.006**	0.266±0.005	0.977±0.001	17810±5038 655±40 (GPU)
	MEGNet	0.157±0.018	**0.223±0.020**	**0.984±0.002**	**5873±66**
	SOAP-SVR	0.916±0.012	1.360±0.004	0.432±0.006	41326±2449
	SOAP-MLP	1.407±0.001	1.762±0.001	0.036±0.002	**85±2 (GPU)**
Formation energy	CGCNN	**0.083±0.007**	**0.140±0.012**	**0.984±0.003**	17865±5217 655±39 (GPU)
	MEGNet	0.134±0.012	0.204±0.017	0.966±0.006	**5870±59**
	SOAP-SVR	0.454±0.004	0.690±0.007	0.617±0.008	19076±1776
	SOAP-MLP	0.892±0.001	1.057±0.001	0.103±0.001	**82±1 (GPU)**
Band-gap energy	CGCNN	**0.387±0.011**	**0.637±0.013**	**0.868±0.005**	17852±5289 694±60 (GPU)
	MEGNet	0.531±0.042	0.814±0.034	0.787±0.018	**5873±53**
	SOAP-SVR	0.843±0.006	1.331±0.017	0.427±0.016	42751±9
	SOAP-MLP	1.326±0.015	1.754±0.019	0.015±0.021	**87±5 (GPU)**
Fermi energy	CGCNN	**0.438±0.009**	**0.657±0.010**	**0.947±0.002**	17928±4844 728±85 (GPU)
	MEGNet	0.604±0.024	0.851±0.030	**0.947±0.002**	**5952±228**
	SOAP-SVR	2.292±0.011	2.836±0.004	0.012±0.005	19789±495
	SOAP-MLP	2.223±0.001	2.759±0.003	0.002±0.002	**85±4 (GPU)**
Bulk modulus	CGCNN	**0.082±0.003**	**0.139±0.015**	**0.825±0.043**	1802±1026 124±14 (GPU)
	MEGNet	0.109±0.010	0.183±0.033	0.738±0.061	**486±15**
	SOAP-SVR	0.286±0.005	0.398±0.018	0.003±0.005	1303±19
	SOAP-MLP	0.275±0.000	0.354±0.000	0.012±0.000	**23±0 (GPU)**
Shear modulus	CGCNN	**0.142±0.004**	**0.287±0.005**	**0.524±0.015**	1746±1010 119±15 (GPU)
	MEGNet	0.170±0.048	0.295±0.080	0.478±0.199	**481±10**
	SOAP-SVR	0.300±0.008	0.422±0.033	0.021±0.002	1181±36
	SOAP-MLP	0.301±0.001	0.460±0.002	0.027±0.007	**22±2 (GPU)**
Poisson ratio	CGCNN	**0.038±0.000**	**0.055±0.001**	**0.400±0.013**	1711±952 115 ± 15 (GPU)
	MEGNet	0.045±0.001	0.064±0.003	0.246±0.037	469±13
	SOAP-SVR	0.057±0.000	0.075±0.000	0.036±0.007	**293±18**
	SOAP-MLP	0.057±0.000	0.074±0.000	0.010±0.003	**21±0 (GPU)**

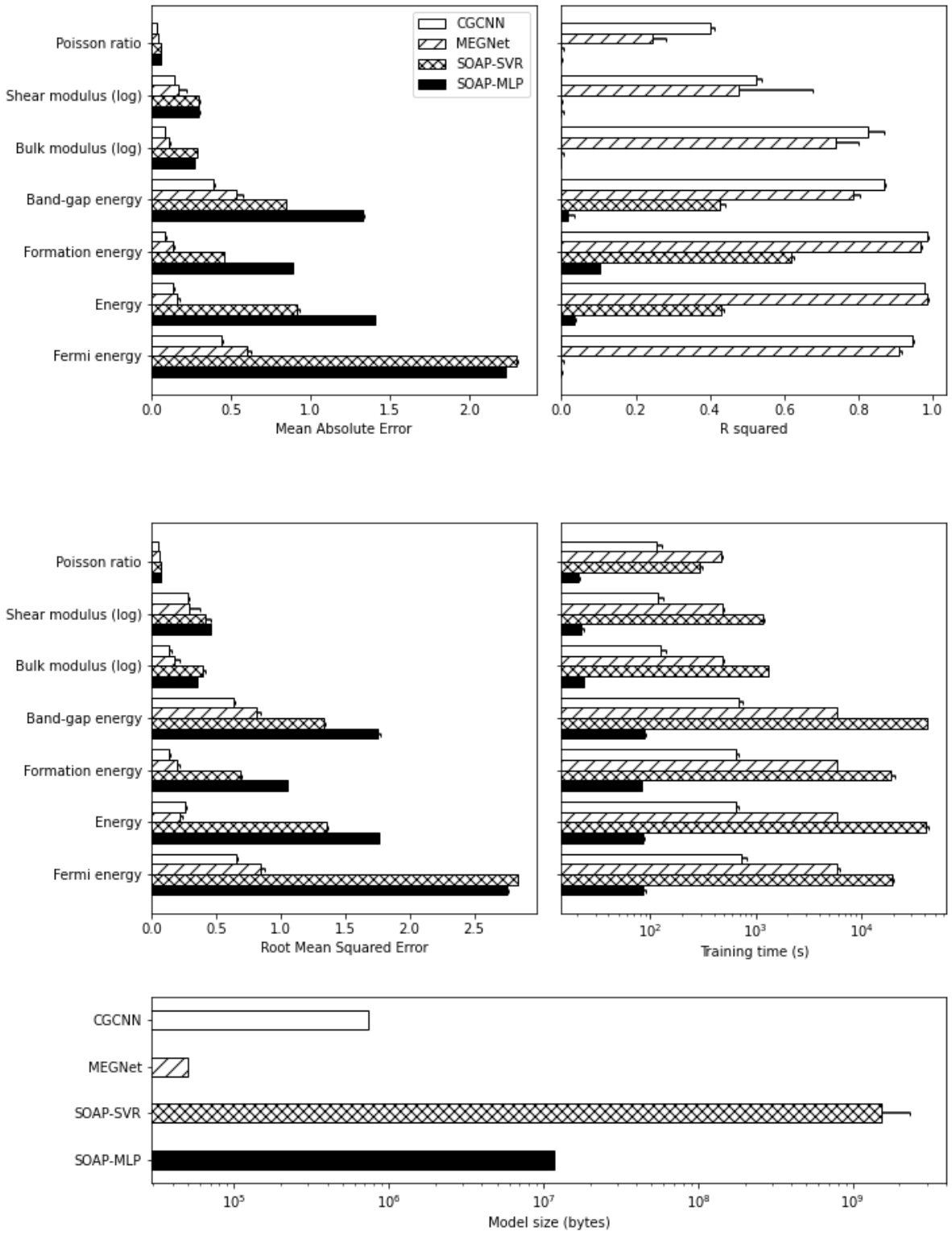


Figure 14. Performance and computational cost metrics for the models under study in this work in the prediction of 7 materials properties. *For a fair comparison, all models were evaluated in CGCNN dataset. For every combination of model and target property, a bar shows the mean value of the corresponding metric after n=10 training rounds (CGCNN and MEGNet) or n=2 rounds (SOAP-SVR and SOAP-MLP). Error bars depict standard deviation. Training time subplot shows best training times for each model (i.e., GPU where possible).*

3.4. GNNs explainability

In section **3.3** of this work, we realized how exact are GNNs in their predictions. Given a graph, first they learn a proper vector representation of the graph and from this graph embedding they predict the desired property. This process occurs with graph convolutions: for each node, information about its neighbours and about itself is aggregated and the vector representation of the node is iteratively updated with the training of the GNN model. Hence, due to the complexity of this learning process, it is practically impossible to know what it is that the GNN has observed in the graph representation of the material's structure to predict a given property.

Recently, interpretability has become particularly important as a part of AI models. If AI models are to be trusted, we must know how or why they make their predictions. Because GNNs lack explainability, several methods have been described recently to address this task (**section 1.3**)**.** Here, **XGNN** [33] has been applied for the **explanation of CGCNN predictions** on six different properties: band-gap energy, Fermi energy, formation energy, bulk modulus, shear modulus and Poisson ratio.

Because XGNN only works for classification tasks, these variables have been **discretized**: values lower or equal than the **median** for each property in the training dataset are class 0 (low), and values above this threshold are class 1 (high). Because of the substantial number of different chemical elements in the materials (87), atoms were masked with their **period** (7 unique labels) and with their **group** in the periodic table (18 unique labels) for the explanations. In the next paragraphs, the results for band-gap energy and formation energy will be described.

A summary of the explanations of XGNN for the predictions of CGCNN on band-gap energy is shown in **Figures 15** and **16**. We can see some trends:

- Chemical elements in groups 15 to 18 are notably more prevalent in explanations for materials with high band-gap energy. The same goes for chemical elements in periods 1 to 3.
- Chemical elements in groups 3 to 12 and periods 4 to 7 are more represented in explanations for materials with low band-gap energy.
- Edges between chemical elements in groups 15 to 18 and periods 1 to 3 are more abundant in explanations for materials with high band-gap energy.
- Edges between chemical elements in groups 3 to 12 and periods 5 to 7 are more significant in explanations for low band-gap predictions.

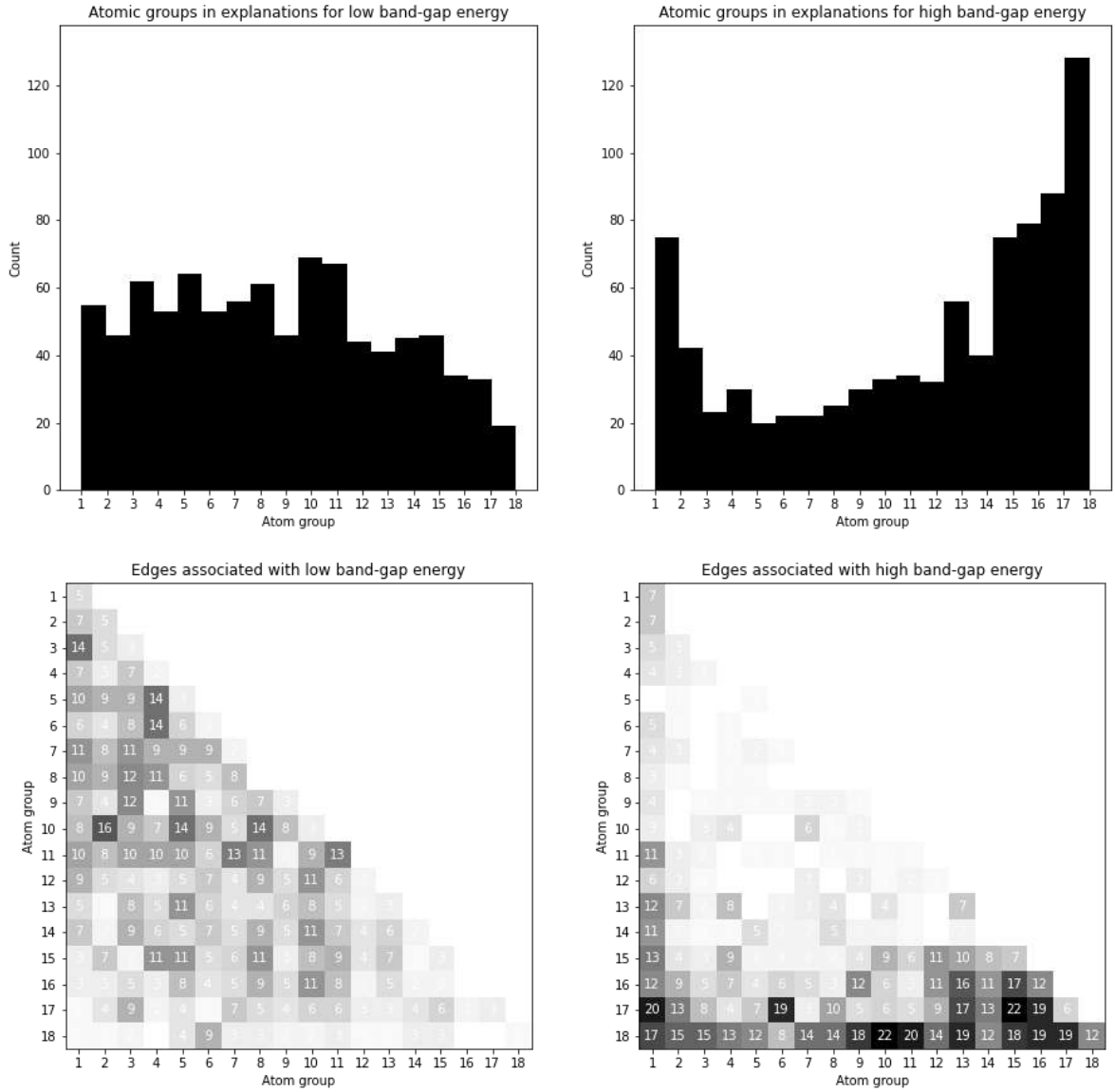


Figure 15. Summary of XGNN explanations of CGCNN predictions on band-gap energy (per atom group). *The presence of metals (groups 1 to 12) and edges between metals are notably higher in explanations for low-energy (conductor materials) predictions, and the opposite for non-metals. In heatmaps, numbers indicate edge counts.*

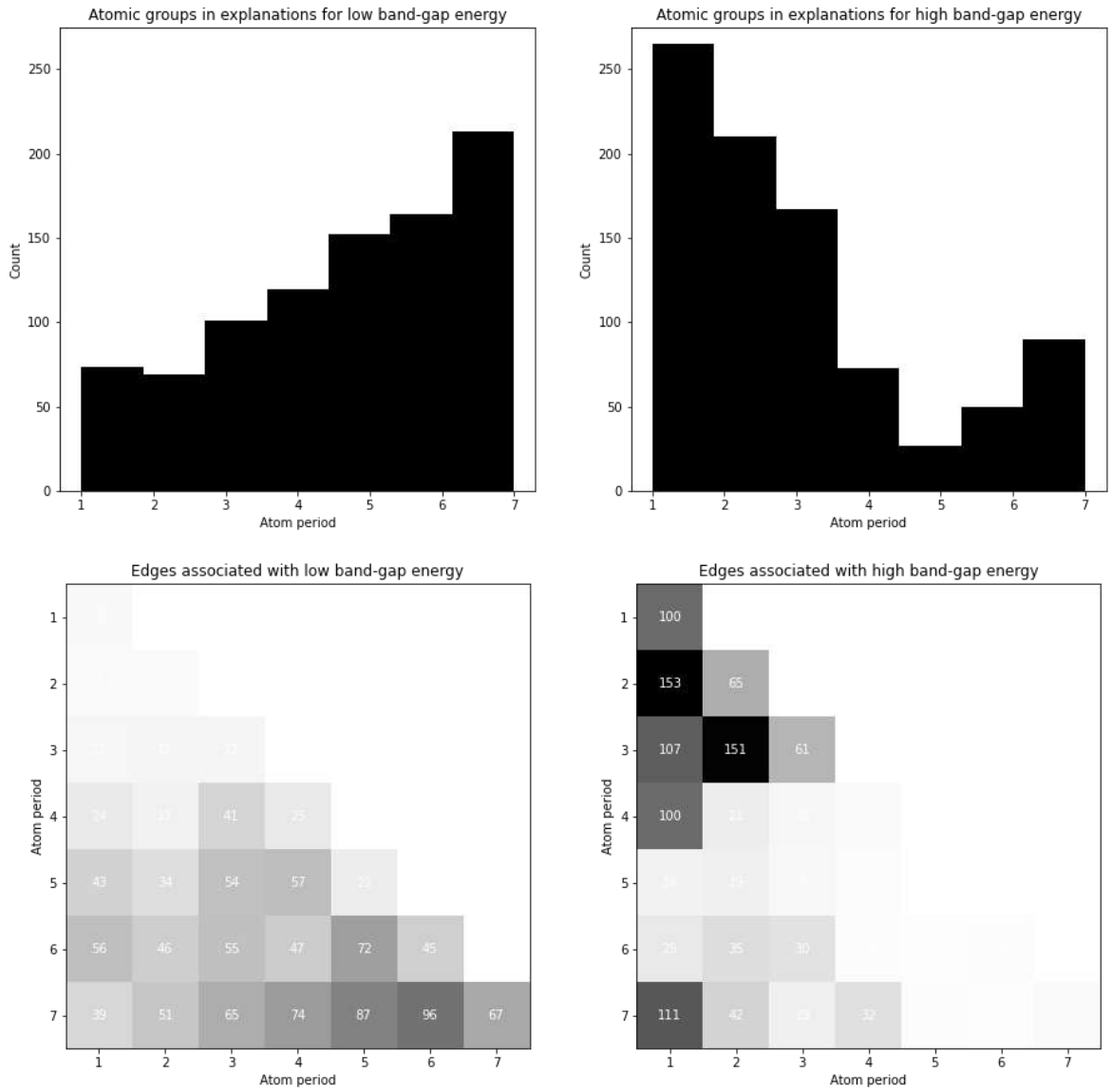


Figure 16. Summary of XGNN explanations of CGCNN predictions on band-gap energy (per atom period). *The presence of metals (periods 5 to 7) and edges between metals are notably higher in explanations for low-energy predictions, and the opposite for non-metals.*

The explanations of XGNN for the predictions of CGCNN on formation energy are summarized in **Figures 17** and **18**. We see that:

- Chemical elements in groups 5 to 15 are more popular in explanations for high formation energy predictions.
- Chemical elements in groups 16 and 17 and period 2 appear much more in explanations for low formation energy predictions.

These results will be discussed in **section 4**.

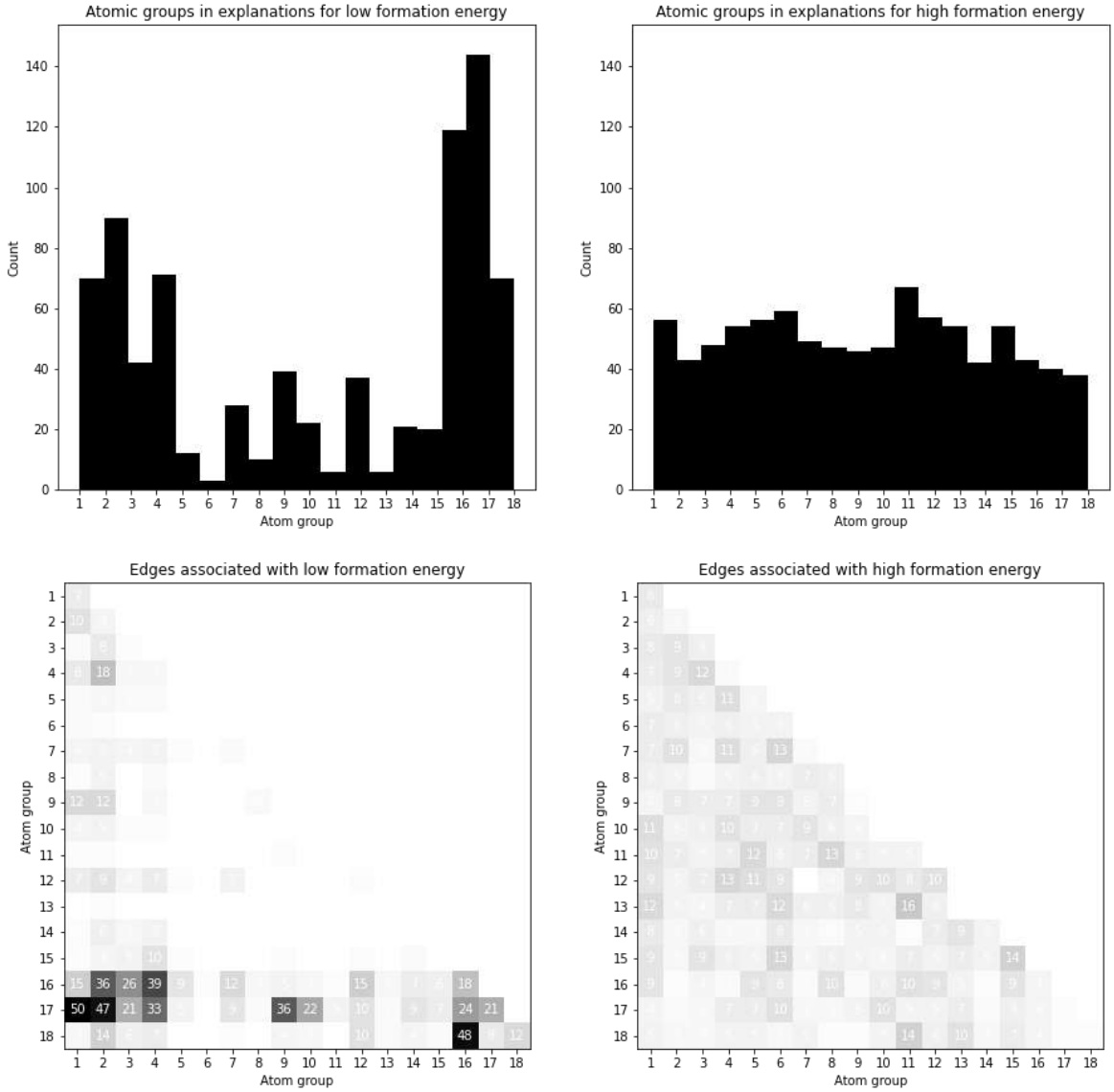


Figure 17. Summary of XGNN explanations of CGCNN predictions on formation energy (per atom group). *The presence of metals (groups 1 to 12) is notably higher in explanations for high-energy predictions, while ionic edges (between atoms in groups 1 to 4 and atoms in groups 16 and 17) are more represented in low-energy prediction explanations.*

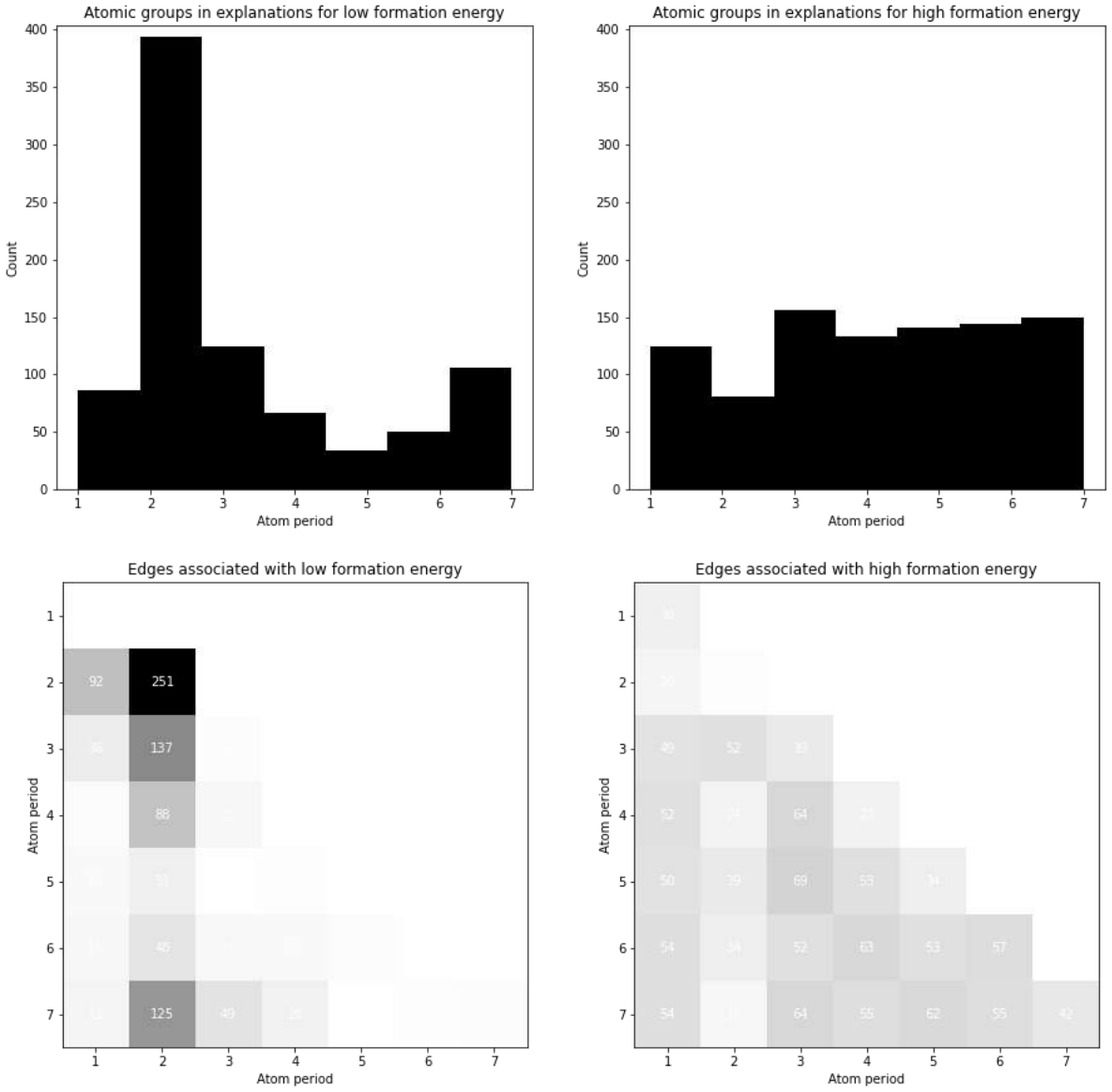


Figure 18. Summary of XGNN explanations of CGCNN predictions on formation energy (per atom period). *Atoms of period 2 are notably more represented in low-energy prediction explanations.*

4. Discussion

LiOn-HD project [7] aims to develop new batteries with improved energy efficiency, lower cost and more respectful with the environment. HI-Iberia [8], the institution where this work was performed, is focused on the cathode of these batteries, which is usually the limiting component of them. Our strategy is to combine generative Deep Learning with a predictive model: while a generator produces novel material's structures, another architecture predicts how the new materials will behave in the batteries' cathode; this predictive model might be a Graph Neural Network. The generated materials with the best predicted behaviour will be tested experimentally in the Instituto de Ciencias Materiales de Madrid (an Institute of CSIC).

In this work, Graph Neural Networks have been used to predict some properties of the materials by looking at their structures (CIF files). To this aim, the Materials Project database [43] has been employed, and some materials properties have been predicted. These quantitative properties are formation energy (normalized by the number of atoms), energy (normalized by the number of atoms), Fermi energy, band-gap energy, shear modulus, bulk modulus and Poisson coefficient. The importance of these properties, whose meaning was described in **Table 2**, is outlined in the next paragraph.

The **formation energy** of a material is proportional to its **stability**: the lower the energy needed to form the material, the more stable it is. The formation energy of a material is equal to its **energy** minus the sum of the energy of its constituent elements alone; then, the energy of a material is also related to its stability. **Fermi energy** is the electrochemical potential of the material at absolute zero; in most cases, it is very close to the **electrochemical potential of the material** at room temperature, and this property could be of interest for predicting the behaviour of a battery cathode [60]. **Band-gap** energy of a material represents the minimum energy that is necessary for exciting an electron to the conduction band and is inversely proportional to **conductivity**. **Bulk and shear modulus** are **elasticity** properties and describe the response of a material to compression and shear stress respectively. The **Poisson coefficient** is another property related to elasticity that gives an idea of the relationship between transverse strain and axial strain when a uniaxial stimulus is applied.

The expressivity of graphs as descriptors was explored in this work. In **section 3.2** was reviewed the method for graph generation from a material's structure of **CGCNN** and **MEGNet** frameworks. In both methods, atoms are nodes in the graph. With the MEGNet approach, edges are added between atoms that are closer than a certain cut-off (hyperparameter). In the CGCNN approach, in addition to this cut-off there is another hyperparameter: for a given atom, only the N closest atoms are considered neighbours (hence, the maximum number of edges for a node is N).

Also, the predictive power of Graph Neural Networks was evaluated. The models employed here were CGCNN [19] and MEGNet [23]. As a benchmark, SOAP descriptors [15] were fed to Support Vector Regressor [27] (SOAP-SVR) and to the region of CGCNN that comes after the graph embedding generation ("L_2 hidden" in **Figure 9B**) (SOAP-MLP). **GNNs notably outperformed** the other two models for all properties predicted in terms of **accuracy** (**Figure 14**). Although SOAP-MLP model training was faster than in the two GNNs, its predictive power was widely worse. Also, model sizes of GNNs were several magnitude orders smaller than those of SOAP models. To sum up, the performance of GNNs was highly superior to that of models not based on graphs in the conditions tested: they achieved a much higher accuracy at a lower computational cost. For a complete comparison, hyperparameter tuning of the four models will be performed in the future.

In recent years, research has put emphasis not only in the predictive power of the ML models but also on explaining why these models make their predictions [28]. In the case of GNNs, they achieve brilliant

accuracy metrics (both regression and classification), but they are so complex that they lack any **explainability**. Traditional methods for the interpretation of DL models might be employed to explain the predictions of GNNs based on node attribute matrices, but they do not bring insights about the structure of the graph. In this work, **XGNN** [33] was applied to explain the discretized predictions of CGCNN in terms of **subgraph patterns** and **atom identities**.

Before diving into the interpretation of **Figures 15** to **18**, it would be convenient to understand what these figures mean. The **histograms** show a count of chemical elements' period or group in the periodic table; these subplots might give insights about **individual elements** associated to a class of materials. The **heatmaps** represent a count of combinations of chemical elements' periods or groups in the periodic table, and this second type of subplot might indicate some **bonds between chemical elements** that are more associated to materials from one class or another. **Metal elements** are widely more abundant in the periodic table in **periods 3 to 7 and groups 1 to 12**, and **non-metals** are more represented in **periods 1 and 2 and groups 13 to 18** [61]. Keeping this in mind, the trends observed in the predictions are discussed in the next paragraph.

The first property whose predictions were explained was **band-gap energy** of a material. As pointed out above in this section, the lower the band-gap energy of a material, the more conductive it is. Hence, we would expect that metals (generally good conductors) have lower band-gap energy than ionic materials. Looking at **Figure 15** and **16**, according to XGNN method, we see that CGCNN has correctly learnt that **metal elements** and **metal bonds** are associated to materials with a **higher conductivity** (lower band-gap).

The other property whose predictions were explained by XGNN was **formation energy** of a material. The lower this property is, the more stable the material. It seems that CGCNN has learnt another trend here: elements from the second period or groups 16 and 17 are heavily related to low formation energy materials. More specifically, bonds between these chemical elements and metals have a higher incidence in low formation energy predictions. This is in accordance with chemical knowledge: **ionic materials and oxides** (bonds between metals and non-metals) tend to be **more stable than metals**.

Some lines of research arise from this work and are being outlined next. The evaluation of the models has not been complete, as default parameters were used for all of them; **hyperparameter tuning** would provide a clearer picture of the actual performance of the four models employed here. For the GNNs to be more robust for their use in LiOn-HD project, some **node or edge attributes of interest in the field of batteries** will be explored and implemented. Finally, a **more complete explanation of GNNs** predictions would be possible if node and edge attributes were considered besides the input graph structures.

5. Conclusion

Graphs as descriptors and Graph Neural Networks as ML models have demonstrated their power in predicting several properties of materials. Applying these frameworks to properties associated with the performance of the materials in the cathode of batteries might boost the discovery of new materials for this application, and the explanation of this predictions with methods like XGNN will provide a more profound knowledge of the chemistry underlying batteries performance.

6. References

(1) Li, J. et al. (2020) AI Applications through the Whole Life Cycle of Material Discovery. *Matter*, vol. 3, no. 2, pp. 393–432.

(2) Jones, O.R. (2015). Density functional theory: Its origins, rise to prominence, and future. *Review of Modern Physics*; 87: 897–917.

(3) Landman U. (1988) Molecular Dynamics Simulations in Material Science and Condensed Matter Physics. In: Landau D.P. et al. (eds) Computer Simulation Studies in Condensed Matter Physics. *Springer Proceedings in Physics*, vol 33. Springer, Berlin, Heidelberg, pages 108–123.

(4) Saxena, A. et al. (2021) Recent advances in materials science: a reinforced approach toward challenges against COVID-19. *emergent mater*. 4, 57–73.

(5) de Leon, N.P. et al. (2021) Materials challenges and opportunities for quantum computing hardware. Science; Vol 372, Issue 6539.

(6) Zitnick, C. et al. (2020). An Introduction to Electrocatalyst Design using Machine Learning for Renewable Energy Storage. eprint arXiv:2010.09435.

(7) HI-Iberia (2021). LiOn-HD. https://www.hi-iberia.es/artificial-intelligence/lion , last consulted: 2021-09-10.

(8) HI-Iberia (2021). HI iberia | Desarrollo software y soluciones TIC. https://www.hi-iberia.es/ , last consulted: 2021-09-10.

(9) A. Géron (2019) The Machine Learning Landscape. In: A. Géron. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition. *O'Reilly Media Inc*., pages: 1–34.

(10) Cognub (2020). Cognitive Platform. http://www.cognub.com/index.php/cognitive-platform/ (via https://towardsdatascience.com/coding-deep-learning-for-beginners-types-of-machine-learning-b9e651e1ed9d), last visit: 2021-03-01.

(11) Kamil Krzyk (2018, July 25). Coding Deep Learning For Beginners. https://towardsdatascience.com/coding-deep-learning-for-beginners-types-of-machine-learning-b9e651e1ed9d , last visit: 2021-09-10.

(12) Dong, S., Wang, P., Abbas, K. (2021). A survey on deep learning and its applications. *Computer Science Review*; 40.

(13) IBM Cloud Education (2020, May 1). What is Deep Learning? | IBM. https://www.ibm.com/cloud/learn/deep-learning , last visit: 2021-09-10

(14) Hoffmann, J, et al. (2019). Data-Driven Approach to Encoding and Decoding 3-D Crystal Structures. eprint arxiv:1909.00949.

(15) Himanen, L. et al. (2020). DScribe: Library of descriptors for machine learning in materials science. *Computer Physics Communications*, 247.

(16) C. Merkwirth and T. Lengauer (2005). Automatic generation of complementary descriptors with molecular graph networks. *J. Chem. Inf. Model*, 45(5): 1159–1168.

(17) F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, and G. Monfardini (2009). The graph neural network model. *IEEE Trans. Neural Netw. Learn. Syst*, 20(1): 61–80.

(18) Qiao et al. (2020). OrbNet: Deep Learning for Quantum Chemistry Using Symmetry-Adapted Atomic-Orbital Features. *J. Chem. Phys*. 153, 124111.

(19) Tian Xie and Jeffrey C. Grossman (2018). Crystal Graph Convolutional Neural Networks for Accurate and Interpretable Prediction of Material Properties. *Phys Rev Lett*; 120(14): 145301

(20) Park, C.W. and Wolverton, C. (2019). Developing an improved Crystal Graph Convolutional Neural Network framework for accelerated materials discovery; *Phys. Rev. Materials* 4, 063801.

(21) Zhou, L. et al. (2021). Machine Learning Enabled Prediction of Cathode Materials for Zn ion Batteries. *Adv. Theory Simul*., 4: 2100196.

(22) Wang, Q. and Zhang, L. (2021). Inverse design of glass structure with deep graph neural networks. eprint arXiv:2104.06632.
(23) Chen, C. et al. (2019). Graph Networks as a Universal Machine Learning Framework for Molecules and Crystals. Chemistry of Materials, 31(9), 3564–3572.
(24) DeCost, B. and Choudhary, K. (2021). Atomistic Line Graph Neural Network for Improved Materials Property Predictions. eprint arXiv:2106.01829.
(25) Yao, Z. et al. (2020). Inverse Design of Nanoporous Crystalline Reticular Materials with Deep Generative Models. *Nat Mach Intell* 3, 76–86.
(26) Bartók, A.P. et al. (2013). On representing chemical environments. *Phys. Rev. B* 87, 184115.
(27) Platt, John (2000). Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. *Adv. Large Margin Classif.*; 10.
(28) Molnar, C. (2019). Interpretable machine learning. A Guide for Making Black Box Models Explainable. https://christophm.github.io/interpretable-ml-book/ , last visit: 2021-09-10.
(29) Ying, R. et al. (2019). GNNExplainer: Generating Explanations for Graph Neural Networks. *Adv Neural Inf Process Syst*; 32: 9240–9251.
(30) Luo, D. et al. (2020). Parametrized Explainer for Graph Neural Network. *Advances in Neural Information Processing Systems*; 33: 19620–19631.
(31) Yuan H. et al. (2021). On Explainability of Graph Neural Networks via Subgraph Exploration. *Proceedings of the 38th International Conference on Machine Learning*, PMLR 139: 12241–12252.
(32) Schnake T. et al. (2020). Higher-Order Explanations of Graph Neural Networks via Relevant Walks. Eprint arXiv:2006.03589.
(33) Yuan H. et al. (2020). XGNN: Towards Model-Level Explanations of Graph Neural Networks. KDD '20: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining: 430–438.
(34) Jain, A. et al. (2013). The Materials Project: A materials genome approach to accelerating materials innovation. *APL Materials;* 1(1): 011002.
(35) Setyawan, W. and Curtarolo, S. (2010). High-throughput electronic band structure calculations: Challenges and tools. *Computational Materials Science*; 49:2: 299–312.
(36) Saal, J. E. et al. (2013). Materials Design and Discovery with High-Throughput Density Functional Theory: The Open Quantum Materials Database (OQMD). *JOM*; 65: 1501-1509.
(37) Talirz, L. et al. (2020). Materials Cloud, a platform for open computational science. *Sci Data*; 7: 299.
(38) NOMAD Encyclopedia (2021). At https://encyclopedia.nomad-coe.eu
(39) Villars, P. et al. (2018). Pauling File - towards a holistic view. *Chem. Met. Alloys*; 11: 43–76.
(40) Allen, F.H. and Shields, G.P. (1999) Crystallographic Databases and Knowledge Bases in Materials Design. In: Howard J.A.K., Allen F.H., Shields G.P. (eds) Implications of Molecular and Materials Structure for New Technologies. *NATO Science Series (Series E: Applied Sciences)*, vol 360. Springer, Dordrecht.
(41) Gražulis, S. (2012). Crystallography Open Database (COD): an open-access collection of crystal structures and platform for world-wide collaboration. *Nucleic Acids Research*; 40:1: 420–427.
(42) Himanen, L. et al. (2019). Data-Driven Materials Science: Status, Challenges, and Perspectives. *Adv Sci (Weinh)*; 6(21): 1900808.
(43) Jain, A. et al. (2018). The Materials Project: Accelerating Materials Design Through Theory-Driven Data and Tools. In: Andreoni W. and Yip S. (eds) Handbook of Materials Modeling. *Springer*, Cham., pages 1751–1784.

(44) Ong, S.P. et al. (2013). Python Materials Genomics (pymatgen): A Robust, Open-Source Python Library for Materials Analysis. *Computational Materials Science*; 68: 314–319.
(45) Ong, S.P. et al. (2015). The Materials Application Programming Interface (API): A simple, flexible and efficient API for materials data based on REpresentational State Transfer (REST) principles. *Computational Materials Science*; 97: 209–215.
(46) You, J. et al. (2018). Graph convolutional policy network for goal-directed molecular graph generation. *Proc. of NeurIPS*: 6412–6422.
(47) Stokes, J.M. et al. (2020). A Deep Learning Approach to Antibiotic Discovery. *Cell*; 180: 688–702.
(48) Jumper, J., et al. (2021). Highly accurate protein structure prediction with AlphaFold. *Nature*; 596: 583–589.
(49) The AlphaFold team (2020, November 30). AlphaFold: a solution to a 50-year-old grand challenge in biology | DeepMind. https://deepmind.com/blog/article/alphafold-a-solution-to-a-50-year-old-grand-challenge-in-biology , last visit: 2021-09-10.
(50) van der Walt, S. et al. (2011). The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science & Engineering*; 13(2): 22–30.
(51) McKinney, W. (2010). Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference*: 56–61.
(52) Hunter, J.D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering; 9(3): 90–95.
(53) Hagberg, A. et al. (2008). Exploring Network Structure, Dynamics, and Function Using NetworkX. *Proceedings of the 7th Python in Science Conference*: 11–15.
(54) Paszke, A. et al. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems*; 32: 8024–8035.
(55) Pedregosa, F. et al. (2011). Scikit-learn: Machine Learning in Python. *JMLR*; 12: 2825–2830.
(56) DScribe Tutorials (2021). Smooth Overlap of Atomic Positions — DScribe 1.1.x documentation. https://singroup.github.io/dscribe/latest/tutorials/descriptors/soap.html , last visit: 2021-09-10.
(57) Xie, T. (2019, October 7). Cgcnn/mp-ids-46744.csv at master · txie-93/cgcnn . https://github.com/txie-93/cgcnn/blob/master/data/material-data/mp-ids-46744.csv , last visit: 2021-09-10.
(58) Chen, C. et al. (2019, July 05). Graphs of materials project. https://figshare.com/articles/dataset/Graphs_of_materials_project/7451351 , last visit: 2021-09-10.
(59) Lee, H. (2016). Density of States, Fermi Energy, and Energy Bands. In Thermoelectrics: Design and Materials, H. Lee (ed.), pages 189–205.
(60) Gao, J. et al. (2015). Brief overview of electrochemical potential in lithium ion batteries. *Chinese Phys. B*; 25: 018210.
(61) Benzhen, Y. et al. (2020) Metals and non-metals in the periodic table. *Phil. Trans. R. Soc. A*.; 378: 20200213.