Masterarbeit
Zur Erlangung des akademischen Grades Master of Science

Predicting Band Gaps and Formation Energies of Solids Using Message Passing Neural Networks

eingereicht von:	Tim Bechtel
Gutachter/innen:	Prof. Dr. Claudia Draxl
	Prof. Christoph T. Koch, PhD

Eingereicht am Institut für Physik der Humboldt-Universität zu Berlin am: 26.10.2022

Abstract

Density Functional Theory is the standard method for *ab initio* computational investigations of novel materials since DFT codes became widely available. However, DFT can be computationally burdensome to investigate a large number of systems (high throughput investigations) or for systems with large number of atoms. As a shortcut to circumvent full electronic structure calculations, machine learning algorithms that benefit from databases of DFT structures have been shown to perform with up to DFT precision when predicting ab initio results. One recent model for atomistic systems are Message Passing Neural Networks, which model the interactions between atoms using an atomistic graph structure. In this work, one molecular dataset and two materials databases are investigated as a testbed for predicting formation energies, electronic band gaps. Additionally, the possibility of uncertainty estimates for single predictions using Monte-Carlo Dropout is tested.

0. Contents

1. Introduction

Traditionally, *ab initio* methods such as density functional theory (DFT)[1] are used in computational materials science in order to explore new materials [2][3][4]. However, it is unfeasible to use DFT on the entire space of possible materials and as such researchers have looked for shortcuts to screen for promising candidate materials such as superconductors or for given applications, such as efficient solar cells. [5][6][7]. Artificial-intelligence and machine learning methods have revolutionised tasks in numerous areas of science and engineering, which have been thought to be impossible or intractable, e.g. image recognition [8], image-to-text translation [9], protein folding [10], quantum chemistry [11], materials science [12], etc. The interest of the materials science community in machine learning techniques is undeniable if one looks at the investments that are made into the research of domain specific models and development of "AI-ready" databases [4][13]. As reference data, e.g. experimental or high-level theory material science data, are still expensive to generate, data efficiency, defined as the ability to generalise to unseen data using only a small amount of training, or known dataset, is also an important concern. One approach to solve this problem is transfer learning [14], which aims to build up general knowledge from a large low-fidelity dataset and then continuously fine tuning the model to capture detail in higher accuracy data.

A recent approach in machine learning has been to include invariances and inductive biases of the training data into the model. For instance, in image recognition, convolutional neural networks exploit the spatial correlation between neighbouring pixels by learning filters that pick out important correlations in neighbouring pixels [15]. A particular challenge for materials science is how to incorporate the oftentimes known geometry of our material in useful way to the model. ElemNet [16] was able to predict properties like bandgap and energy per atom quite well without any information about the geometry/structure of a material but only about what elements are used to create this material. Feature engineering which uses descriptors such as SOAP, Coulomb Matrix, attempts to include more structural information into the model [17]. The SOAP matrix encodes local atomic environments by expanding atomic pairs in radial basis functions and spherical harmonics and calculating the overlap of neighboring atoms. The Coulomb Matrix describes the pairwise correlations of atoms using a Coulomb potential based on the interatomic distance. Recently, graph neural networks have appeared as a promising method where we can input the material or chemical structure to the model. This allows the model to distinguish two materials with the same stoichiometry (chemical formula).

This thesis presents work using graph neural networks, specifically message passing neural networks. We first discuss regression generally in terms of statistical learning. We then describe convolutional neural networks in terms of additional symmetries of the data. Then, we move on to the desire for rotational invariance in physical systems and how graph neural networks can achieve this invariance. The final message passing neural network is then evaluated on three different datasets. The first two of which, QM9 consisting of small organic molecules and the Materials Project database consisting of existing as well as hypothetical inorganic crystals, serve to compare the model against literature. The third dataset, AFLOW which contains bandgaps and formation enrgies of DFT calculations on crystal systems is used as a new testbed for evaluating inferences on crystal structures.

2. Machine learning models

2.1 Types of machine learning tasks

Different types of machine learning can be charaterized as supervised and unsupervised learning and regression and classification tasks. With supervision, the model learns to predict output features from an input datapoint as a mapping $f : X \to Y$ that maps $x \in X$ to $y \in Y$. This is often done by defining a loss function $\mathcal{L}(y, \hat{y})$ that is minimal when the prediction $\hat{y}$ is equal to the target output y. When the target values y are single or multiple scalars this can be seen as (multiple) regression. On the other hand, if y is a class label, i.e. a discrete variable, then the prediction $\hat{y}$ is often the estimated probability of different classes, and the problem becomes one of classification. In unsupervised machine learning there are no labels attached to the input, and the goal might be to e.g. find out different classes or subsets in the input space. Another task might be to generate new data which is in the input domain of the data that the model learns on [18] or to learn a compressed representation of the input data [19]. There are also intersections between supervised and unsupervised tasks where datasets are partially labelled [20] or the input features are reconstructed from injected noise [21]. These methods deal with incomplete, noisy data in order to improve prediction accuracy. This work focuses on supervised prediction of graph labels, i.e. single scalars, using graphs as input/training data.

2.2 Neural networks

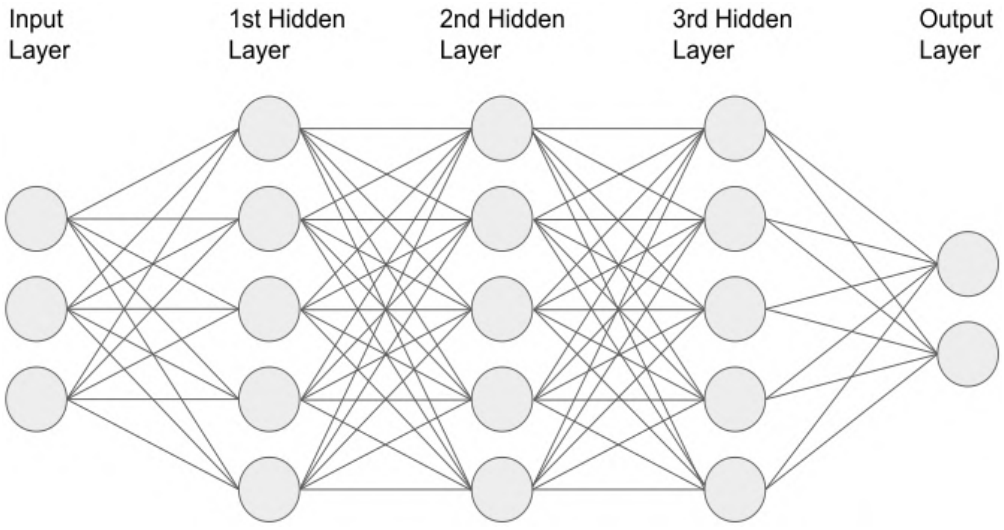

Figure 2.1: Visualization of a neural network, with three input nodes, three hidden layers with five neurons each, and two output nodes. Fully connected layers connect the different layers of the network.

In order to understand Graph Neural Networks, one first needs to introduce Neural Networks, which form a basic building block of most state-of the-art models for learning on graph topologies and modeling the interactions between nodes in a graph. More precisely Artificial Neural Networks, hinting at the inspiration from biological neurons, are computational models that learn from data

and have the ability to generalize, to some extent, to new data. This is done by applying learnable linear transformations and non-linear activation functions on the input vector successively.

This paradigm of training models to perform complex tasks, such as character recognition is in contrast to the previous hand engineered features used for such tasks [15]. The model learns from the input data by calculating a forward pass, in which the input is passed through the model successively, as shown in Fig. 2.1, while the activation of each hidden layer and output is saved. The three-dimensional input is passed from the three input nodes into the first hidden layer by multiplying with a 3×5 weight-matrix, adding a bias term and applying a non-linear function, resulting in a 5-dimensional hidden representation. If the weight-matrix is non sparse, this layer is called a fully connected layer. This is repeated for the 2nd, 3rd, and the output layer, resulting in a two-dimensional output. In the supervised learning setting the output of the model, which we will call the prediction, is compared to the target of the input data, and their deviation is quantified in a cost- or loss function which is defined as:

$$C(w, b) = \frac{1}{N} \sum_{x} (y(x) - \hat{y}(x, \{w, b\}))^2 \tag{2.1}$$

Here, w and b are the learnable weights and biases of the model, $y(x)$ is the target scalar for input x, and $\hat{y}(x, \{w, b\})$ is the prediction of the model for input x, given its weights and biases. The summation goes over all inputs x in the dataset of size N.

The act of training the model on data can then be defined as minimizing the loss function for all examples in the dataset. This would ideally be done with gradient descent through back-propagation, but due to computational constraints such as limited memory and a sometimes unstable gradient leading to an ill-converged model, most often stochastic gradient descent is used, where a small batch of data is fed into the model one by one [22]. This also prevents the optimiziation from getting stuck in local minima and in Eq. 2.1, in order to normalize the loss function, N becomes the number of samples in one batch, N_{BS}.

2.3 Convolutions

Another key concept in Graph Neural Networks is the convolution. Convolutional Neural Networks (CNNs) make use of the fact that relevant features in the input, which could e.g. be an image, are translationally invariant. For example, if the task is to detect whether a dog is present in an image, it is not important where in the image the dog is located. This translational invariance is exploited through spatial filters, so called convolutions. A single convolution in the sense of CNNs for images i.e. two-dimensional data, maps a region of pixels to one output value corresponding to the location of the pixel-region, by taking a weighted sum of the neighboring pixel values (see Eq. 2.2).

$$x'_{ij} = \sum_{n=-1}^{1} \sum_{m=-1}^{1} w_{nm} x_{i+n, j+m} \tag{2.2}$$

The weights of this sum are shared for a single filter which means that it is parameter efficient, and they can be learned through backpropagation [9]. For image recognition often multiple filters are learned to pick out different features in the image. Each of the k filters are swept across the 2D input image using a specific stride length and for each point the convolution is calculated. The result is one 2D output for each filter. The side length of the output data depends on input data size, filter size and stride length. Using multiple convolutions in conjunction with pooling layers and non-linearities, Deep Convolutional Neural Networks can be built up, which for a long time have been state-of-the-art in image recognition [8].

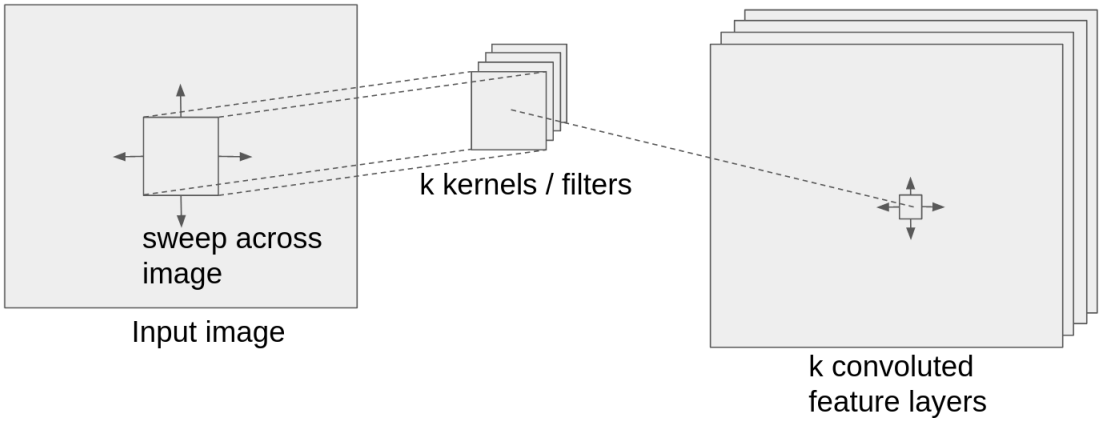

Figure 2.2: Visualization of a convolution operation with k filters and 2D input data, resulting in k convoluted feature layers.

2.4 Graph neural networks

Graph neural networks are a relatively new development in the field of deep learning. They first have been introduced as a generalization of recurrent neural networks, which have been primarily designed to work on sequences. This first model [23] was tested on graph theory tasks and showed the possibility to learn from local neighbourhood interactions on graphs. Subsequent research went in two directions. On the one hand multi-task models have been developed that perform well on all kinds of tasks from graph theory through computer vision to chemical predictions [24]. On the other hand, very problem-specific models are developed in the chemical and materials science fields, e.g. DimeNet [25] and NequIP [26], that exploit additional constraints imposed on the graphs and their properties. These models are still accurate given comparatively few datapoints for training.

2.4.1 Invariances in training data

Certain invariances in the training data can be exploited to improve training and generalization capability of a model. In CNNs, parameters are shared in single convolutional layers, thus making the model more parameter efficient [15], and in turn reduce overfitting, which generally increases with the number of parameters.

Convolutions can not only be defined for two-dimensional data structures, but also for arbitrary topologies when a local neighborhood can be defined. This is the case for graphs, since they are defined by nodes that interact pairwise with other nodes through edges, which allows us to abstract complex relational structures as objects and their pairwise relations [27]. To draw another analogy to image convolutions, just as a convolutional filter acts on a local patch of a two-dimensional datastructure, i.e. 3-by-3 pixel grid within an image, a graph convolution updates a node based on the neighboring nodes and the connecting edges to them. In the possibly simplest graph convolution, the central node is updated by adding the average feature of the connecting nodes. This results in something like a diffusion model on the graph. On this relational inductive bias, very general, yet accurate models can be built.

The important symmetries of a physical system, under which the model should be invariant, are translational-, rotational- and permutation symmetry. Translational and rotational invariances are given if the representation only includes pairwise distances, thus a global translation of positions does not change the graph representation. These two symmetries are given by isotropy of space in which the physical system is embedded. Permutational invariance is wanted if the system represents e.g. a collection of atoms because similar atoms should be indistinguishable, and the order in which the atoms are listed should not change the system representation or model output. Another strategy to induce symmetries within the model is data augmentation. In this method, training samples are transformed in a way in which the input changes significantly, but the output target stays the same [28]. This relates back to the idea of the target being invariant under certain transformations. With these augmentations, the network is trained to learn an invariant mapping

from input to target even though the invariance is not inherently given by the architecture of the model.

Even though these two strategies can be seen as orthogonal, they are often used in parallel to reduce the number of parameters and expand the training data at the same time, to produce state-of-the-art results [29]. In conjunction with an auxiliary loss, adding noise to graph representations (similar to augmenting the graph training samples) can work as a powerful regularization tool that enables much deeper networks to be trained effectively [21].

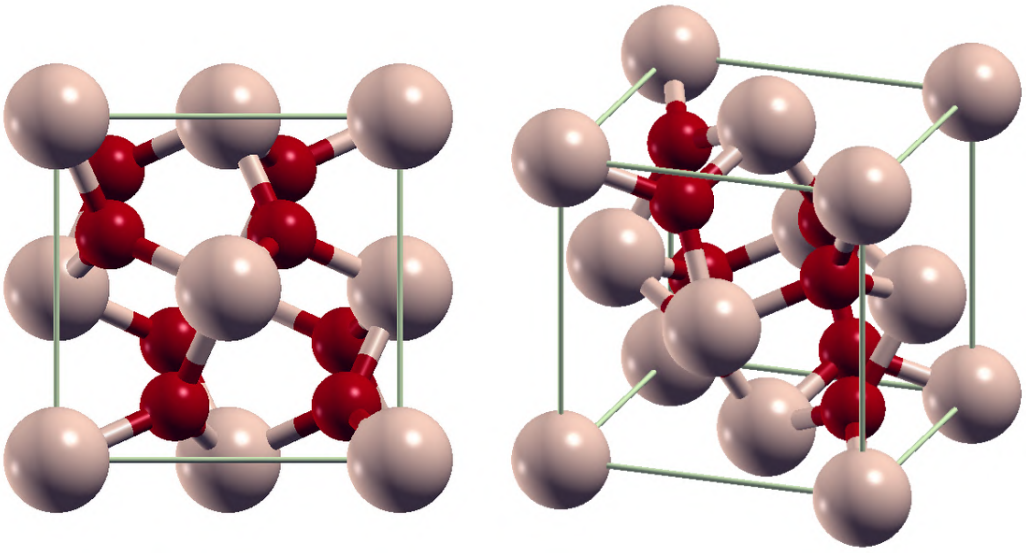

Figure 2.3: Two 2-dimensional projections of the same 3-dimensional structure. The differing angle results in two very different images of the structure. The material is SiO2 in space group number 205, with two atoms in a face-centered cubic (fcc) lattice.

2.4.2 Graph representation

In order to explain specific models that operate on graph structured data, the basics of graph theory and representation need to be defined. A graph is defined as a set of nodes V (for vertex, being equivalent to nodes), corresponding edges E, and a global state of the graph, u:

$$G = (V, E, u) \tag{2.3}$$

Nodes in the graph have features vectors $h_i \in R^F$, and N nodes can be concatenated into the node feature matrix, H:

$$H = \{h_1, h_2, ...h_N\} \in \mathbb{R}^{N \times F} \tag{2.4}$$

The edges of graph G are defined by the adjacency matrix A:

$$A_{ij} = \begin{cases} 1 & \text{if node } i \text{ is connected to node } j \\ 0 & \text{if } i = j \text{ or otherwise} \end{cases} \tag{2.5}$$

The neighbourhood of node i is defined as the set of all neighboring nodes:

$$N(i) = \{j \mid A_{ij} = 1\} \tag{2.6}$$

The global state u can also be called label, e.g. if we use this attribute for regression. It can have any number of dimensions: $u \in \mathbb{R}^{F_u}$, but in the following sections we will usually refer to it as a scalar. This state describes the graph as a whole, and plays the central role when predicting graph-level features.

2.4.3 Graph convolutional neural networks

The first graph neural networks were refined by the development of a graph convolutional network which takes more inspiration from convolutional neural networks [20]. This model was then trained on real world data, namely citation and knowledge graphs, to predict node labels, while only a small amount of labels could be used for supervised learning. The node update equation for a graph convolutional layer l is defined as:

$$H^{(l+1)} = \sigma(\widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \tag{2.7}$$

Here $H^l \in \mathbb{R}^{N\times F}$ describes the node feature matrix in layer l with N nodes in the graph and F features per node. $\widetilde{A} = A + I_N$ is the adjacency matrix with added self connections, $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$ is the degree matrix, which counts the number of connections of each node and $W^{(l)}$ is a trainable weight matrix for layer l. σ denotes an activation function such as $ReLU(\cdot) = max(0, \cdot)$. This simple update layer is able to propagate information through the graph along the edges with every step, and thus model interactions between nodes. The functional form is inspired by spectral graph convolutions. This convolution smoothes label information out over the graph and is used to classify unlabelled nodes, resulting in a model where connected nodes via edges are seen as having similar labels, i.e. the adjacency matrix encoding node similarity.

This notion is however not true when modeling interaction between atoms which might be very different, e.g. hydrogen and silicon which have 1 and 14 electrons, respectively. Even though these two elements are close in a material and therefore share an edge (see Section 4.1), they are not at all similar. What this tells us, is that an extension or generalization of the simple Graph Convolution is needed.

2.4.4 Message passing neural networks

Further generalization of the graph convolutional model resulted in the development of message passing neural networks [11]. This model now includes edge features between adjacent nodes and the concept of a message being passed between nodes that depends on the information in both connecting nodes and the edge feature. With the ability to approximate complex interactions between nodes, the model was used to fit quantum chemistry calculations on the QM9 dataset [30].

The so-called message M_{ji}, with sending node j and receiving node i, is an edge-wise vector of length F_m. As every node can have different numbers of incoming and outgoing messages, for every node these messages need to be aggregated in a size- and permutation-invariant way. This is usually done by summing or taking the average of incoming or outgoing messages, but other methods are possible and sometimes beneficial, as shown in [24]. We call the aggregated node-wise message for node i, m_i:

$$m_i^{t+1} = \sum_{j\in N(i)} M_{ji}(h_i^t, h_j^t, e_{ij}^t) \tag{2.8}$$

The node-wise aggregated message is then used in the node-update equation for layer t and node i

$$h_i^{t+1} = S_t(h_i^t, m_i^{t+1}). \tag{2.9}$$

The edges can also be updated in a similar way as detailed in [31], depending on the previous edge vector and the connecting, updated nodes

$$e_{ij}^{t+1} = E_t(h_i^{t+1}, h_j^{t+1}, e_{ij}^t) \tag{2.10}$$

This updating of edges is not necessary to build a message passing model, but it enables the model to learn different interactions, depending on the latent representations of nodes and edges. After T steps of message passing, the node features are aggregated to produce a predicted global feature $\hat{y}$, using a readout function:

$$\hat{y} = R(\{h_i^T \in G\}) \tag{2.11}$$

This global feature can have one or more dimensions, depending on the regression or classification task. When there are multiple possible targets for regression in a single graph, like e.g. molecular properties, it is usually beneficial to train one model for each property, at the expense of computation time [25] [21].

2.5 Related work

There are numerous variants of how the general graph network as defined in [27] can be implemented, and many different ways of making use of the capabilities of GNNs. In order to explain the choice of model used in this thesis, some related models are introduced. The most closely related implementations of the message passing neural network are SchNet [12], CGCNN [7] (architecture shown in Fig. 2.5), which also predict energies of formation and electronic band gaps but use discrete edge attributes and no edge update function. The MatErials Graph Network (MEGNet) presents a simple framework that incorporates not only edge updates but also global input properties and global updates, with streamlined update functions. An illustration of this model is shown in Fig. 2.4. The model surpasses the preceding CGCNN and SchNet in predictive performance on the QM9 and Materials Project dataset, but is slightly lower performing than the MPNN with edge updates in [31]. The advantage of the MPNN model with edge updates could be an effect of the improved update function with elementwise multiplication, reminiscent of ResNet [32]. The importance of choosing the right update function is also shown using CGCNN [7], where two update functions are tested on otherwise equivalent models.

Other approaches use transfer learning to train a model on a large but less accurate dataset and then fine-tune it to be accurate on smaller but more accurate data, like experimental material science data [14]. All of these approaches have the potential to apply well in other domains while the understanding of each component can still be improved.

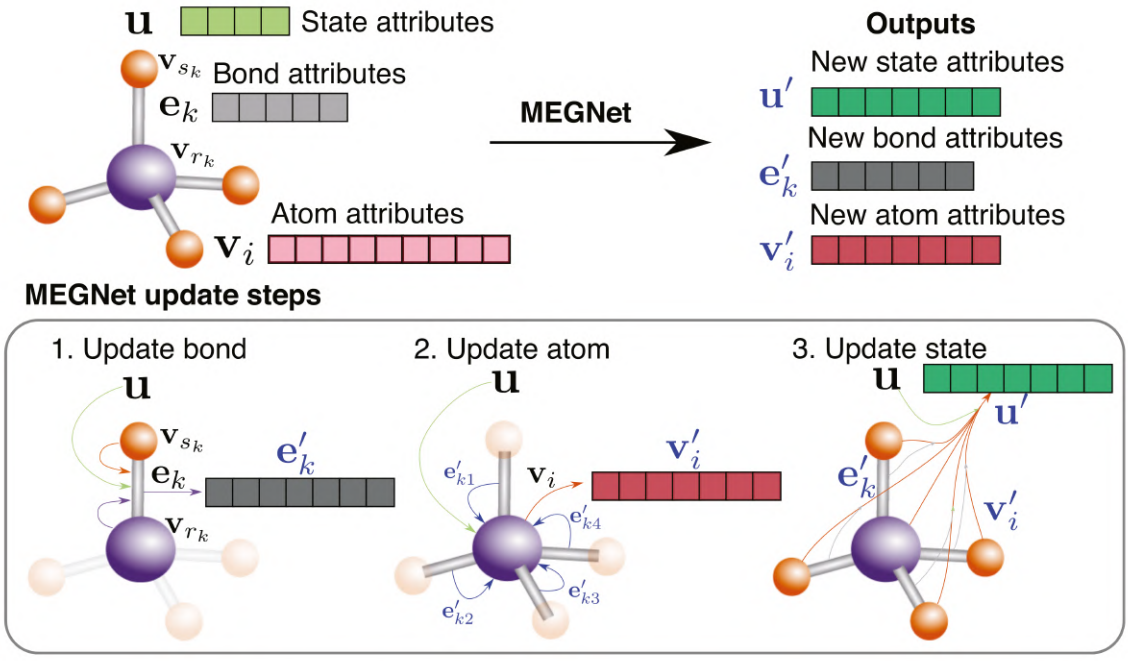

Figure 2.4: Overview of a MEGNet module as taken from [33] with permission. The graph represents a methane-like structure. The graph has node attributes $V = \{\mathbf{v_i}_{i=1:N}^{v}\}$, edge attributes $E = \{(\mathbf{e_i}, r_k, s_k)_{i=1:N}^{e}\}$ (with receiver node r_k and sender node s_k), and global state attributes $\mathbf{u}$. This visualizes the iterative updating of graph features as described in the message passing algorithm.

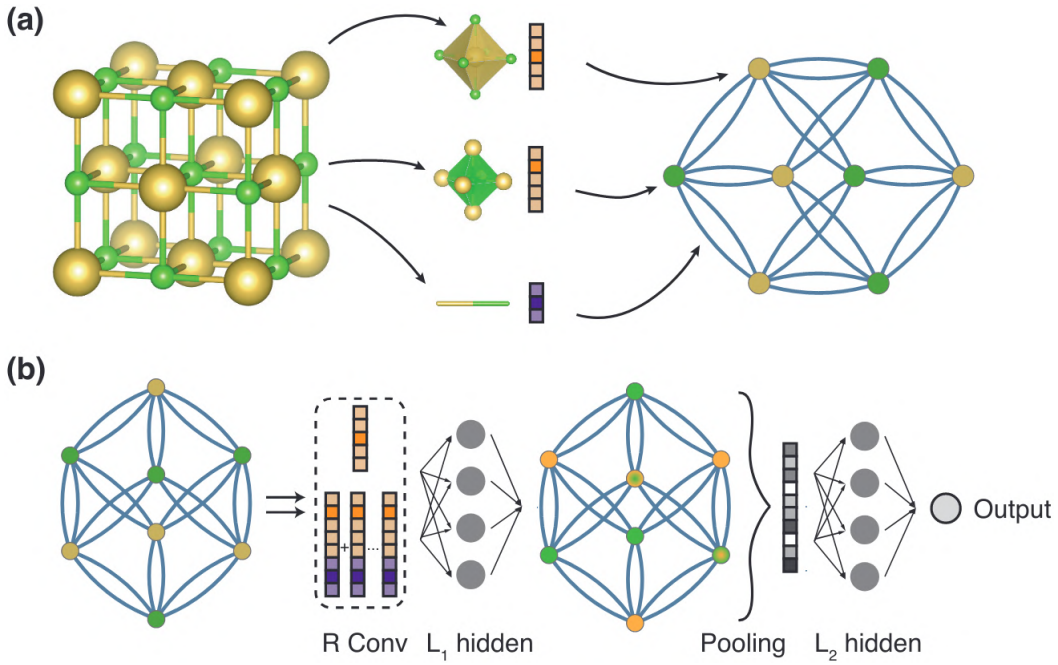

Figure 2.5: Illustration of the crystal graph convolutional neural network (CGCNN) from [7], taken from the publication with permission. (a) Construction of the crystal graph. Crystals, i.e. periodic structures, are converted to graphs with nodes representing atoms in the unit cell and edges representing atom connections. Nodes and edges are characterized by vectors corresponding to the atoms and bonds in the crystal, respectively. (b) Structure of the convolutional neural network on top of the crystal graph. R convolutional layers and L_1 hidden layers are built on top of each node, resulting in a new graph with each node representing the local environment of each atom. After pooling/aggregation a vector representing the entire crystal is connected to L_2 hidden layers, followed by the output layer to provide the prediction.

3. Datasets

3.1 Quantum Machines 9

The QM9 database, as it is often referred to in the literature, has 133,885 molecules with up to nine atoms (C, O, N, F) besides hydrogen. The QM9 database was published to enable benchmarking and testing of new and existing methods [30]. All structures have been relaxed and properties such as energies, harmonic frequencies, and dipole moments have been calculated at the B3LYP/6-31G(2df,p) level of quantum chemistry. The variety of different properties also uncovers shortcomings of specific models that otherwise might not have been noticed [33][21]. This dataset has been used in numerous publications on graph neural networks and can thus be cross referenced and used as comparison with many other models. [25][11][31][21].

3.2 Materials Project

The Materials Project database ("MP-DB" in the following, in order to distinguish from "message passing" as "MP") is a core program of the Materials Genome Initiative, and aims to develop a platform for accelerating materials design by making high-throughput materials science computations open-source [34]. The database contains compounds based on the Inorganic Crystal Structure Database (ICSD) [3], together with hypothetical compounds. Relaxed geometries and energies are calculated using the Vienna Ab initio Simulation Package (VASP) [35]. For a subset of calculations, other properties like elastic, or thermoelectric properties are computed [36]. The MP-DB is continuously updated with new materials and corrections to old calculations, so that the number of compounds (state October 2022) is 146k. Some older literature in machine learning still worked with the database when it contained only 70k compounds [31][33].

3.3 AFLOW

The ALOW database is also based on the mostly experimentally reported structures in the ICSD, along with binary-alloy cluster expansions and hypothetical compounds [36]. Properties reported are relaxed geometries, energies, band structures, bulk- and shear moduli, thermomechanical properties, etc. in different subsets of the dataset [5]. Standards for the underlying calculations within the AFLOW repository are described in [37].

Structures are either taken from the ICSD database or generated from prototype crystal geometries and then relaxed using consistent convergence parameters such that the forces on each atom are below a certain threshold. After obtaining the relaxed structures (if the material did not come from the Binary Alloy database), two more calculations are performed: STATIC and BANDS, the first of which is the basis for thermodynamic and electronic properties in AFLOW, and the second generates the electronic band structure.

Then the parameters for k-point sampling, potentials and basis sets, Fourier transform meshes, DFT+U, etc., corrections are chosen based on predefined rules that take into account the contributing elements and the dataset from which the structure came. If a calculation fails due to hardware constraints or at runtime, the parameters are adapted. The electronic band gap in the AFLOW repository is obtained by taking the difference between the conduction band minimum and the valence band maximum, using the band structure calculated along high-symmetry paths

in reciprocal space. [38]

The samples used in this work were pulled from the AFLOW repository using its REST API and querying for materials where both the enthalpy of formation per atom and the electronic band gap have been calculated, along with all the parameters that are used to construct the relaxed unit cell used in the DFT calculation. The calculations were also filtered to only use calculations with the "PAW-PBE" DFT-type label. This resulted in 102,428 entries (as pulled in June 2022). As no structure had external pressure applied (pV=0), the values labelled as enthalpy of formation in AFLOW are equivalent to energy of formation and therefore comparable to the MP-DB. Energies of formation are a widely used benchmark to test machine learning models for materials science [7][31][33][14][39] since energies can be interpreted in an atomistic way, with a specific contribution from each atom and its local neighborhood. Also, the energy of formation at zero temperature (T=0K) and no applied pressure can be used in conjunction with the convex hull to assess the stability of a given phase of a crystal in comparison to other competing phases [38].

3.4 Comparison of materials databases

As described before, both materials databases (excluding the molecules dataset QM9), use VASP as the underlying software package. The pseudo potential method is employed by projector-augmented-wave (PAW) with the Perdew-Burke-Enzerhof (PBE) [40] parameterization of a generalized-gradient approximation (GGA) to the DFT exchange-correlation functional [36]. Although both databases use the same DFT methods, the parameters used in the individual calculations are different, e.g. choice of pseudopotential, which value to use as elemental reference state for formation energies, and when to use DFT+U correction. One such parameter is the plane wave energy cutoff, whose effect can be seen in differences in relaxed unit cell volumes, when comparing AFLOW, OQMD and MP-DB structures. The energy cutoff for AFLOW is the highest with 560 eV, while MP-DB, and OQMD have cutoffs of 520 eV and 400 eV, respectively. This parameter choice also affects band gaps and hints at the fact that the band structures might be more converged in the AFLOW dataset and therefore more consistent overall, leading to a better model.

Differences in the reference chemical potential of oxygen also contribute to deviations in energies of formation between the different databases, but when training on a single dataset, this offset for oxides should be easily learned by the model. Another parameter is the choice of PAW pseudopotential for different structures. VASP makes different choices with respect to pseudo potentials with changing settings in the high-throughput databases. The potential has minimal effect on formation energies, but the band gap values are affected severely. Some of these parameters will be investigated in how they affect the prediction performance of the GNN.

4. Methods and Implementation

In the following section, the explicit methods for generating graphs is introduced along with the equations that define the message passing algorithm and how the network is implementend in code. The code is available at [41]. In the end of this chapter, the benchmark results using similar datasets as in [31], QM9 and MP-DB, are discussed.

4.1 Representing systems of atoms as graphs

As discussed in Section 2.4.1, graphs allow us to construct a representation of a physical system that has the proper invariances, and in section 2.4.2 this graph data structure was described. In this section, we will define the physical system that we want to represent, and how we embed this in a graph structure.

In materials science, an ideal material (or crystal) can be described just using a single unit cell and defining the periodic boundaries [42]. This includes the lattice vectors and the positions of N atoms within the unit cell. The lattice vectors define the tiling of space and can be grouped into one of the 14 Bravais lattices. Every crystal in three-dimensions has three lattice vectors $\mathbf{a_1}$, $\mathbf{a_2}$, $\mathbf{a_3}$, which span the primitive (unit) cell and can be defined in terms of their lengths and pairwise angles. When we produce linear combinations of the lattice vectors with integer multiples, every point R in the lattice can be reached (Eq. 4.1),

$$\mathbf{R} = n_1\mathbf{a_1} + n_2\mathbf{a_2} + n_3\mathbf{a_3}, n_i \in \mathbb{N} \tag{4.1}$$

After defining the lattice vectors, in order to characterize the full crystal, a basis of N atoms is needed. The basis describes the positions of the atoms within the primitive cell, and it can consist of any number of atoms. Examples for different basis are elemental crystals with only a single atom as the basis, like Cu, Ag or Au, the binary material NaCl has a basis of two atoms, while protein crystals may consist of thousands of atoms per unit cell. When the atomic positions are given with periodic boundariy conditions (PBC) or without PBC, then a local neighborhood can be defined for each atom. This is usually done by calculating the distance to every other atom within a specified cutoff radius. When there are PBCs, this can result in one or more neighbors being in the periodic images of the unit cell, i.e. the distance vectors between pairwise neighbors going across unit cell boundaries. Specifically, for smaller unit cells, if the cutoff is larger than the unit cell length (e.g. for cubic unit cells), the atoms neighbor themselves, possibly multiple times. This reflects the periodic boundaries of the system in the resulting graph, even though the graph is not explicitly periodic. The adjacency matrix (see Eq. 2.5) is then defined by the neighborhood, in contrast to Eq. 2.6, where the neighborhood is defined by the adjacency. In fact, it is not possible to construct the adjacency matrix as defined in Eq. 2.5 for periodic systems, as there can be multiple edges for each pair of atoms, and the representation is not considered a simple graph with at most one edge per pair of nodes, but it is considered a multi-graph.

Apart from the constant cutoff, which is an intuitive way to construct the atomic neighborhood, it can also be constructed using the k-nearest neighbors (KNN). As the name suggests, only the k nearest atoms are defined as the neighborhood of each atom. When the cutoff is a constant length, the resulting graph is symmetrical, whereas for KNN the graph is not symmetrical but the number of neighbors is constant. This has several benefits. Firstly, there are no isolated atoms without neighbors, where the neighborhood is undefined and which could make the prediction unstable or undefined. Secondly, it was hypothesized that a constant number of neighbors makes the message

passing algorithm more stable in general [31]. Lastly, the dynamic batching scheme (explained in Section 4.5) is less likely to fail if the number of edges is bounded by the number of nodes ($N_{edge} = kN_{node}$, for KNN), however, this is only a problem for small datasets with a large variance in graph sizes. It has been shown in [31] that a KNN cutoff with $k = 24$ neighbors produces the lowest mean absolute error when predicting formation energies on the OQMD dataset. So this neighborhood cutoff will be used in this work, going forward, but if faster gradient steps are desired or if the network architecture is changed significantly, this cutoff should be re-evaluated.

4.2 Message passing implementation

In the following sections, the specific functions used for graph updates will be defined, as taken from [31]. We found this to be a good benchmark network, that achieved state of the art results when it was introduced, while still being relatively simple to implement.

4.2.1 Node-/Edge embedding

Before nodes are updated, the raw node and edge features have to be embedded. The atomic number (per node) and pairwise distance (per edge) is transformed into representations that the neural network can learn from. With class labels this is usually done by one-hot encoding (OHE). The atomic species of atom i, Z_i, is one-hot encoded into a vector with the number of different atomic species as length. This OHE is then transformed into a vector with latent size F_N by multiplying with a trainable weight matrix W_0, as follows:

$$h_i^0 = W_0 \cdot \text{OHE}(Z_i) \tag{4.2}$$

The edges are embedded using a radial basis expansion, where the pairwise distances d_{ij} between atoms is described using Gaussians,

$$(e_{ij})_k = exp\left(-\frac{(d_{ij} - (-\mu_{min} - k\Delta))^2}{\Delta}\right), k = 0 \dots k_{max} - 1. \tag{4.3}$$

This dimensional expansion from a scalar to a vector of size k_{max} is analogous to one-hot encoding, in that the model can decorrelate the input and output more easily [31].

4.2.2 Node-/Edge update functions

The nodes and edges are updated at each message passing step t. First the edge update is calculated and applied, and then the node update is applied using the updated edges. For each edge, an edge-wise message M_{ij} is defined, that connects node i to node j. The equation for M_{ij} is:

$$M_t(h_i^t, h_j^t, e_{ij}^t) = M_t(h_i^t, e_{ij}^t) = (W_1^t h_i^t) \odot \sigma(W_3^t \sigma(W_2^t e_{ij}^t)). \tag{4.4}$$

The $\odot$ denotes element-wise multiplication. This function can be seen as a continuous filter, where the edge feature attenuates the node feature, after both have been transformed by feed-forward layers. The aggregation of edge-wise messages is defined as follows:

$$m_i^{t+1} = \sum_{j \in N(i)} M_t(h_i^t, h_j^t, e_{ij}^t). \tag{4.5}$$

Edge-wise messages are aggregated into node-wise messages by either taking the sum of neighboring features (as in Eq. 4.5) or the mean of neighbors. The edge update equation is defined as follows:

$$e_{ij}^{t+1} = \sigma(W_{t+1}^{E2} \sigma(W_{t+1}^{E1}(h_i^{t+1}; h_j^{t+1}; e_{ij}^t))) \tag{4.6}$$

Eq. 4.6 consists of concatenating the sending and receiving node for nodes i and j with the edge feature e_{ij}, which connects the two nodes. This concatenation is then passed into a two-layer neural network with two shifted soft-plus activation functions. Nodes are then updated with the node update equation:

$$h_i^{t+1} = S_t(h_i^t, m_i^{t+1}) = h_i^t + W_5^t \sigma(W_4^t m_i^{t+1}). \tag{4.7}$$

Eq. 4.7 uses the aggregated messages m_i^{t+1} and the original node features h_i. The node-wise message is transformed in a two-layer neural network with one activation function and added with the node feature to gain the updated node h_i^{t+1}. This addition with the previous feature has similarities to the residual connections used in ResNet [32], which enables researchers to train deeper convolutional networks than without the residual connections.

4.2.3 Global readout function

After T message passing steps, the interactions between nodes are stopped and the graph feature is read by aggregating the node features to a single scalar. This is done by transforming the node features in a neural network with one activation function and a hidden size of $C/2$ and subsequently summing over all nodes in the graph or taking the mean:

$$R(\{h_i^T \in G\}) = \sum_{h_i^T \in G} W_7 \sigma(W_6 h_i^T) \tag{4.8}$$

Whether the sum or the average is taken over all nodes (in Eq. 4.8 depends on the dataset and the fitted property, here we show the equation for summation. This aggregation has to be invariant with respect to the permutation of nodes, as their ordering should not matter and it also different numbers of nodes have to be aggregated, as the graphs can have variable sizes.

Other aggregation functions for the readout have been tested and notably the set2set function resulted in the best performance in [11]. set2set is based on the hidden state of a recurrent neural network and can thus be expected to pick up on more complicated relations in a set of nodes. However, this aggregation function is not evaluated here, since it adds more hyperparameters and thus increases the training and evaluation complexity.

4.3 Libraries / Frameworks

As a computational framework, the *JAX* ecosystem [43] is used because of its use in recent state-of-the-art research [21]. Features like automatic gradients and just-in-time compilation, and active development that foster new scientific discoveries are especially appealing. In conjunction with *JAX*, we use *Haiku* [44] for trainable neural network layers and *Optax* [45] for optimization routines. Finally, for general graph networks we use *Jraph* [46] which provides a functional API to apply transformations to arbitrary graphs. These four libraries form a cohesive framework for the whole process of training a machine learning model (preferably neural network), apart from the database interface.

As a local database for atomic structures the *Atomic-Simulation-Environment* database (ASE-DB) is used for its straight forward selection and manipulation of compounds from the database. The conversion of atomic structures to graphs is done only once for each database as described in Section 2.4.2, using a constant cutoff or k-nearest neighbors, therefore the time consuming graph generation does not have to be repeated for each hyperparameter experiment.

4.3.1 Complete algorithm

All of the single functions, like node update function and readout function are defined independently in separate *JAX* functions. They are then put together inside multiple *GraphNetwork* functions provided by *Jraph* which perform the algorithm on the whole graph. This allows the whole algorithm to be just-in-time compiled by *JAX*, which provides a significant speedup over non-compiled functions.

Algorithm 1 Message passing algorithm with edge updates

function GNN(V,E) ▷ First, embed edges and nodes
  for all $d_{ij} \in E$ **do**
    $e_{ij}^0 \leftarrow RBF(d_{ij})$ ▷ expand distances in radial basis functions
  end for
  for all $Z_i \in V$ **do** ▷ Loop over atomic number of nodes in graph
    $h_i^1 \leftarrow W_0 \cdot \mathrm{OHE}(Z_i)$ ▷ One hot encode atomic numbers and multiply by weight matrix
  end for
  $t \leftarrow 0$ ▷ Initialize layer counter
  while $t \leq T$ **do**
    $e_{ij}^{t+1} \leftarrow E_t(h_i^{t+1}, h_j^{t+1}, e_{ij}^t)$ ▷ Update edges with edge update function
    $M_{ij} \leftarrow M_t(h_i^t, h_j^t, e_{ij}^t)$ ▷ Calculate edge-wise messages
    $m_j^{t+1} \leftarrow \sum_{i \in N(j)} M_{ij}$ ▷ Aggregate edge-wise messages to nodes
    $h_i^{t+1} \leftarrow S_t(h_i^t, m_i^{t+1})$ ▷ Update nodes with incoming messages
    $t \leftarrow t + 1$
  end while
  $\hat{y} \leftarrow \frac{1}{N_N} \sum_{h_i^T \in G} NN(h_i^T)$ ▷ Apply Neural net on nodes and take mean over all nodes
return $\hat{y}$
end function

4.4 Optimizer

For optimizing all message passing neural networks in this thesis, the Adaptive Moment Estimation (ADAM) optimizer [47] is used. The ADAM optimizer combines adaptive learning rate and momentum updates [48] in order to efficiently find global or at least good local minima in the landscape of the loss function with very high dimensional models. The adaptive learning rate is important in the case of our heterogeneous materials dataset, as some features of the data, i.e. atomic number, are sparse, since some atomic species are very rare compared to others in the dataset. Momentum in this context means that the present gradient step also takes into consideration the previous gradient step. Momentum is used to avoid oscillatory behavior in regions with steep gradient curve in one dimension and a flat gradient other dimensions.

The update rule for Adam is written as:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \tag{4.9}$$

where η is the learning rate, $\hat{m}_t$ and $\hat{v}_t$ are bias corrected first and second moment estimates of the gradient, which are updated in the following way:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{4.10}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{4.11}$$

Common values for β_1 and β_1 are 0.9 and 0.999, which are also used in the implementation of this thesis.

Despite the adaptive learning rate and momenta built into Adam already, most publications use a decaying learning rate and exponential moving average in addition to the optimizer. This is mostly a heuristic strategy, but it has proven to be a reliable strategy for making training more stable and improving convergence of the training and evaluation losses. Many different learning schedules are used in the literature like cosine decay, warm up steps, exponential decay, or a combination of two. For simplicity and in order to stay close to the reference implementation in [31], an exponentially decaying learning rate is used in this work. This mimics annealing the parameters and avoids jumping between local minima at the end of the training process.

4.5 Training loop

The training loop defines how the weight matrices are updated, when the model is evaluated on the training dataset, when training is stopped, and when the state is checkpointed. Before the training of the model, the data is divided into training, validation, and test data in an 80:10:10 split. Training and validation data are used for cross validation, and the model is finally evaluated on the test data to asses the model's performance on samples it has not encountered.

The model is trained with dynamic batches of a maximum of $N_{Batch} = 32$ graphs (including a padding graph) using the ADAM optimizer [47] provided by Optax. Batches are sampled without replacement from the training dataset and reshuffled every epoch. Dynamic batches are created by calculating the average number of nodes and edges for $N_{Batch} - 1$ graphs and rounding this result up, in this case to the next multiple of 64. This value (power of 2) is motivated by the processor architecture that is used, in that CPUs are often 64 bit and GPUs use banked memory and specific optimized kernels that work best with data sizes of 2^N.

Then, during the training loop, graphs are sampled without replacement from the training dataset, until the maximum number of nodes or edges is reached, or $N_{Batch} - 1$ graphs are retrieved. The rest of the budget is then used for padding, and the result is a static number of nodes, edges, and graphs. This only needs to be just-in-time compiled once and therefore greatly increases the speed of each graph network evaluation.

During each training step, the gradient of the loss function with respect to the model weights W_n is computed, and the model weights are updated according to the gradient and the ADAM optimizer state and momenta. The loss function is the mean-squared-error:

$$\mathcal{L}(\{x_i\}, \{W_n\}) = \frac{1}{N_{Batch}} \sum_{i=1}^{N_{Batch}} (\hat{y}(x_i) - y_i)^2 \tag{4.12}$$

Here, x_i is the input to the model, i.e. the crystal graph, $\hat{y}(x_i)$ is the predicted scalar of the model for input x_i, and y_i is the true label for the graph i. Every $50,000$ gradient steps the model is evaluated on all splits, but only the validation error is used for model selection. During evaluation, the mean-squared-error (MSE) and the mean-absolute-error (MAE) are evaluated, and if the validation MSE decreases, the model is saved as the new best model.

5. Results

In this chapter, the results of the graph neural networks are described. First the model is evaluated on two benchmark datasets QM9 and MP-DB, as to include both molecules and periodic structures for reference. The comparability of these results is discussed with respect to other literature on this data. Secondly, we present promising results on the AFLOW dataset, both when fitting energy of formation and the electronic bandgap, followed by evaluation on unseen test data. These results are compared to a descriptor based model and the advantages of GNNs are discussed.

5.1 Learning internal energy on QM9

The procedure for normalizing the inputs is the same as described in [31]. Though the target energy is the total for the whole molecule, we want the model to fit energy contributions of each atom, as this fits closest with the architecture of the model, with respect to updating the nodes and edges iteratively. To achieve this, the mean energy per atom for the whole dataset is calculated by $\hat{\mu} = \frac{1}{N}\sum_i \frac{t_i}{n_i}$, where t_i is the target energy and n_i is the number of atoms in the i-th sample, N is the number of samples in the whole dataset. Then the standard deviation of the mean energy per atom is calculated as: $\hat{\sigma} = \frac{1}{N}\sqrt{\sum_i (\frac{t_i}{n_i} - \hat{\mu})^2}$. Finally the targets are rescaled be subtracting n_i-times the mean energy per atom and dividing by the standard deviation: $\tilde{t_i} = \frac{t_i - n_i\hat{\mu}}{\hat{\sigma}}$. The scaled targets are used to train the model, so the model also outputs scaled predictions, which need to be scaled back, by multiplying by standard deviation and adding the atom-wise mean. This step is implied in all following regression tasks.

The model performance on the internal energy as obtained from the QM9 dataset can be seen in Fig. 5.1. As shown, the energy regression is precise with respect to the standard deviation of the data, with a coefficient of determination of $R^2 > 0.999$ on the test set. On this dataset and model, overfitting on the training data is not too prominent but still observable, when looking at the MAE on training data compared to the MAE on the test data (8.7 meV compared to 13.9 meV). As discussed in Section 4.5, we define the optimization as minimizing the MSE, i.e. the gradient is calculated using the MSE, which is a smooth function at the origin, while the MAE is not smooth at zero error. The reason for choosing the MSE for the loss function and gradient is being able to compare with other literature on QM9, but heuristically, the MSE should also weight outliers in the target distribution more as the gradient of the MSE increases towards higher errors, while the gradient of the MAE is flat over the whole target range.

When comparing to the original results for this model architecture, it is obvious that the MAE on unseen data (test split) in this work is around 30% larger than in [31]. This can have various reasons, like different molecules in each split, especially in the test split. Another possibility is the different batching. In [31], the batching was done statically, i.e. with one batch size for the whole training loop, while in this work, we use dynamic batching, where the batch size is adapted to the size of the included graphs. These differences in the training/evaluation paired with errors of the same magnitude in our implementation and that of [31], suggests that the re-implementation is successful.

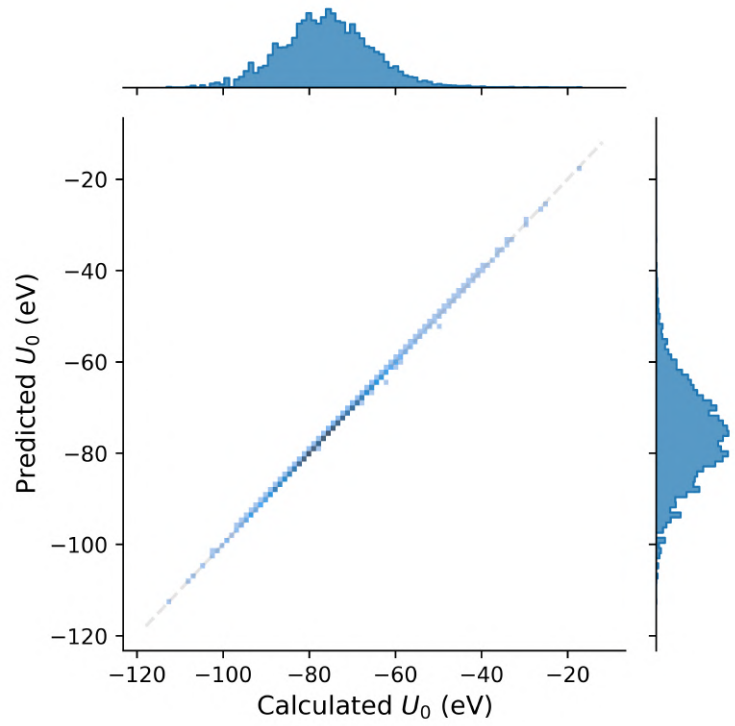

Figure 5.1: Model prediction for the internal energy U_0 from the QM9 dataset, using the base model on the test split. The blue squares represent histogram bins.

5.2 Learning formation energy on MP-DB

When trying to test the model on data from the MP-DB [49], we are facing not only the problem of unavailable split data, but also that the database has expanded since the publication of the reference model [31]. Nonetheless, we do the same filtering for elements, excluding noble gases (He, Ne, AR, Kr, Xe) and get 84 different elements, similar to the reference. For our dataset, this results in a total of 126,147 materials, divided into a 80:10:10 split. In comparison, the reference dataset from the MP consisted of 69,640 compounds, so nearly half as many. In theory, nearly doubling the dataset size should help the model with generalization and improve the overall test error, but it is unclear whether the additional data come from the same distribution.

We compare the regression models by fitting formation energies, that have been normalized in the standard way by subtracting the mean and dividing by the standard deviation of the targets and after model inference, the targets and predictions are scaled back to the original distribution. This produces a model with an MAE of 42 meV/atom on the test dataset, with a coefficient of determination $R^2 = 0.994$. As the accuracy of DFT predictions compared to experimental results (assumed ground truth, although there can be significant variance between different experiments) is around 100 meV/atom [50], the model's fit of DFT data can be considered relatively precise. But when comparing this MAE to the reference result of 22.7 eV/atom [31], our model is not as accurate, not taking into account the different underlying datasets, which is hard to correct for.

From the learning curve (in Fig. 5.3), we can also see that the model stops improving after fewer than 1 million gradient steps, on the validation and test data. This is undesirable and hints at tuning the learning rate (which was taken from the reference implementation) or adding regularization (not included in the reference model), as the training error continues improving even after 1 million gradient steps. Evidently, the model has the capability to adapt to the training dataset as the training loss decreases steadily, but it lacks generalization capability. This is likely due to a relatively high number of parameters (around 2.5 million weights) without a regularizing loss.

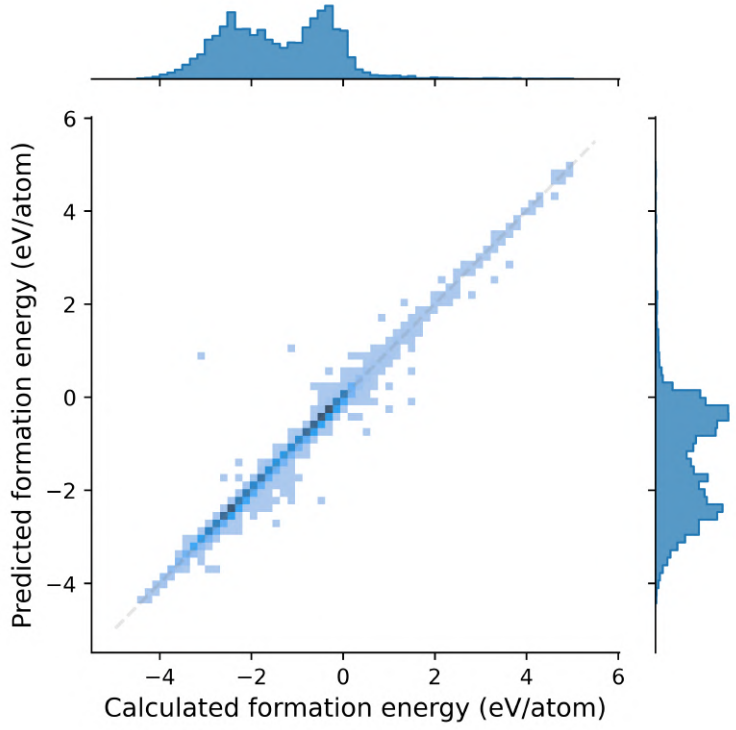

Figure 5.2: Model predictions for formation energy per atom on the Materials Project database, using the test split. The dashed grey line is the ideal fit $\hat{y}(x) = y$. Every blue square is one histogram bin.

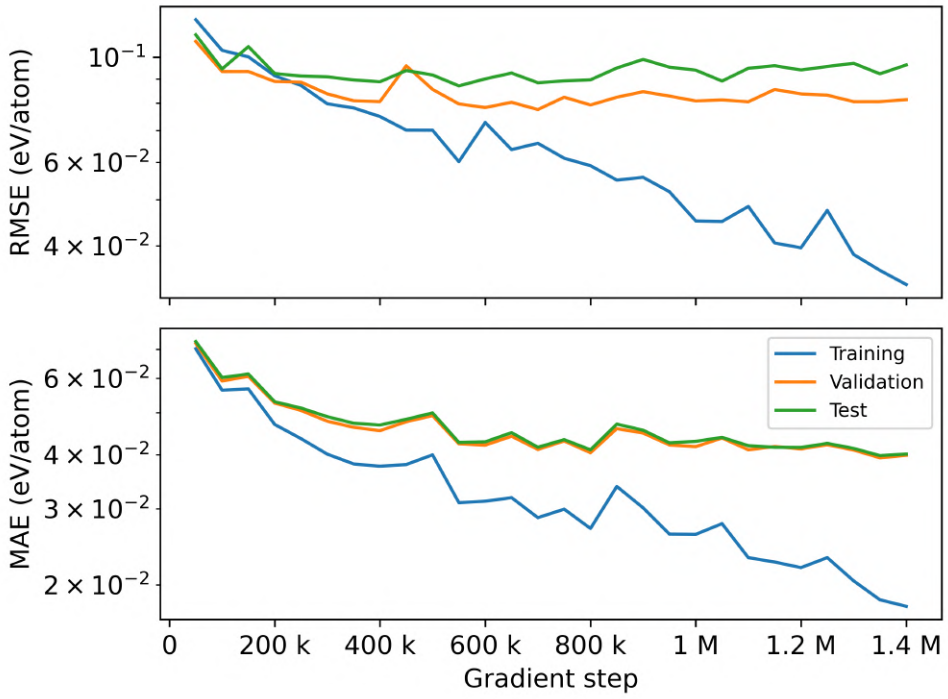

Figure 5.3: Learning curves for fitting the reference model on Materials Project data. Root mean squared error on the top, mean absolute error on the bottom for all three splits. The training is stopped when the validation RMSE does not improve in the last one million gradient steps. Here, this is the case at step 1.4 million. The model with the lowest validation RMSE (here, at step 700,000) is saved and used for evaluation on the test split.

5.3 Learning formation energy on AFLOW

In order to evaluate this model on untested data from the ALOW database, we pull the data as described in Section 3.3. The target values are normalized as described before in Section 5.2 using the mean and standard deviation. Outliers with a formation energy of less than -10 eV/atom or higher than 70 eV/atom were removed. This is equivalent to removing outliers of more than 5 σ away from the mean. Also, only calculations using the PAW-PBE DFT-type label are used to be more consistent. Calculations with other functionals are present in AFLOW, but only represent a small fraction of the data. With this filtering, the relatively heterogenous dataset is made slightly more homogeneous.

The regression of formation energies results in a fit with an MAE of 19 meV on the test set, with a coefficient of determination of $R^2 = 0.991$, which is more precise than the fit of the MP-DB. The regression result is shown in Fig. 5.4, along with the learning curve in Fig. 5.5. This improved result on the AFLOW materials can be interpreted as an indicator of the heterogenity of the Materials Project data, while the ALFOW data is more homogenous. This can be a result both from tighter convergence of the calculations or from more similar input structures in AFLOW, i.e. fewer different atomic species (84 in MP-DB, after removing noble gases, versus 74 in AFLOW, including noble gases). Another observation can be made when grouping the structures with their respective prediction error into one of seven crystal systems. It can be seen that the error is lowest for cubic systems in the test split. This might be expected, since around 87% of the dataset consists of cubic systems. However, it also confirms that the model learns from the graph topology and not only from pairwise distances and thus predictions including their precision is heavily influenced by graph connectivity. This is facilitated by including next-neighbor interactions with multiple message passing steps.

Model	**Property**	**MAE (meV)**	**RMSE (meV)**	$\mathbf{R}^2$	**Accuracy**
MPEU	E_F /atom	24 ± 13	69 ± 16	0.993 ± 0.004	-
MPEU	E_{BG}	251 ± 28	524 ± 54	0.917 ± 0.021	0.984 ± 0.003
PLMF [5]$*$	E_{BG}	350	510	0.90	0.93

Table 5.1: Summary of tenfold cross-validated models on formation energies (E_F) and band gaps (E_{BG}). ($*$ indicates an earlier AFLOW dataset snapshot). The model in this work is denoted as MPEU (message passing with edge updates). The last column shows accuracy of the logistic regression applied to the band gap predictions, and the other metrics on E_{BG} refer to predicted non-metals. Error values are calculated from the standard deviation over ten random training-validation-test splits.

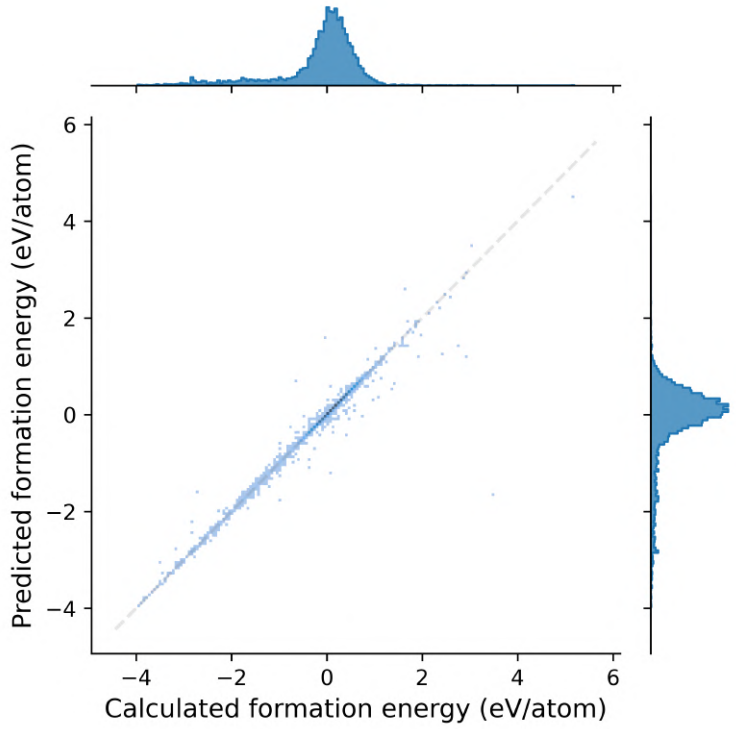

Figure 5.4: Predicted versus DFT energy of formation for compounds from the AFLOW database, using the test split. This model resulted in the best performance from the cross-validated hyperparameters. The dashed line shows $\hat{y}(x) = y$, i.e. where points of a perfect fit would lie.

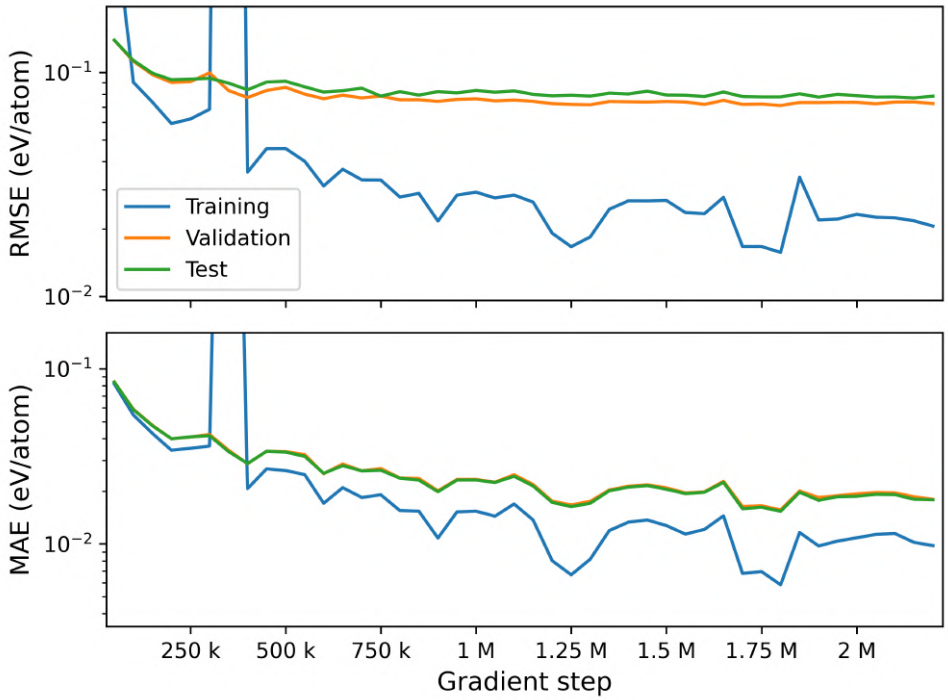

Figure 5.5: Learning curves for different splits when fitting the AFLOW energies of formation per atom, evaluated using root mean squared error (RMSE) and mean absolute error (MAE). The outlier in the training split likely results from a bad gradient step that produces a highly inaccurate prediction on a single datapoint. In this case, the optimizer still recovered from this gradient step.

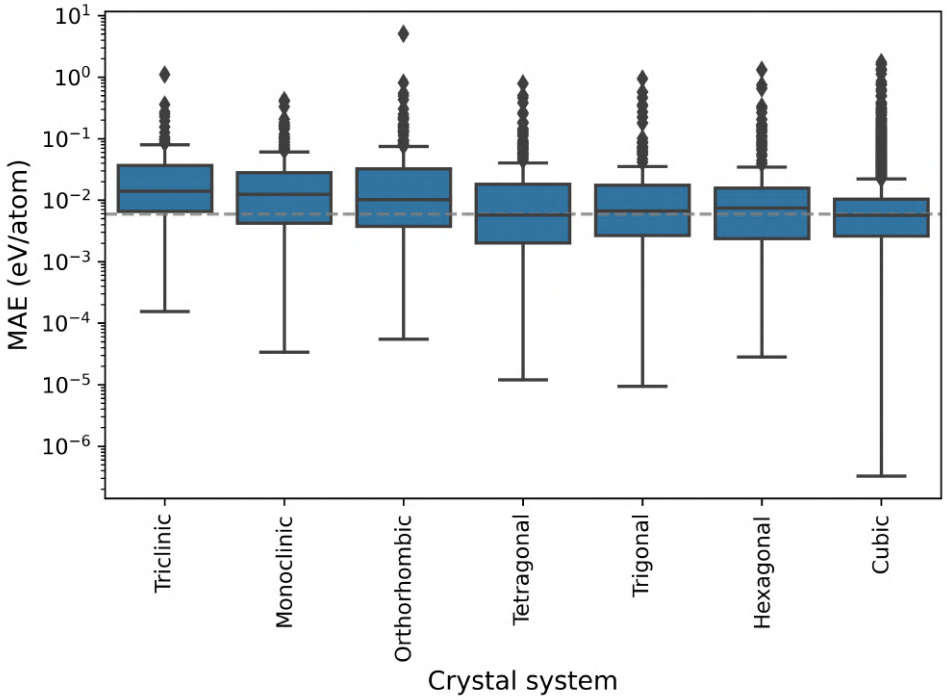

Figure 5.6: Box plot of absolute prediction errors for formation energy per atom with respect to the crystal system. The dashed grey line shows the median error of all classes, as a guide to the eye.

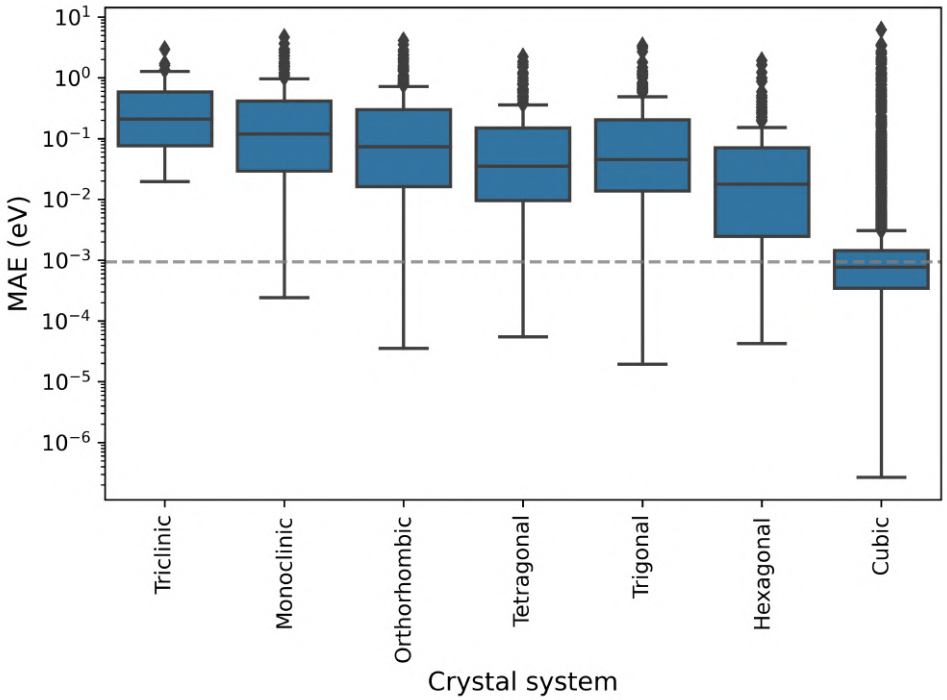

Figure 5.7: Box plot of absolute prediction errors for band gap with respect to the crystal system. The dashed grey line shows the median error of all classes, as a guide to the eye.

5.4 Learning electronic band gap on AFLOW

Predicting the electronic band gap is expected to be a more challenging task than energies of formation. Formation energies can still be understood as having contributions from each atom and their bonds, whereas the band gap cannot be estimated from contributions of single atoms. The property "electronic band gap" is calculated in reciprocal space, which challenges the basic architecture of GNNs that act on local neighborhoods within the atomistic graph. Additionally, in terms of density functional theory, the property that is actually calculated in the materials databases is the difference in Kohn-Sham energies, which is systematically different from the true band gap of a structure. Even if the exact functional was used in the DFT calculation, the calculated Kohn-Sham band gaps are not equal to the true band gaps, whereas the formation energy would be exact. However, this does not prevent us from learning Kohn-Sham band gaps, and thus, this property can still serve as a useful benchmark.

Another indicator for the difficulty of predicting band gaps is the scaled MAE for the E_g and the E_f benchmarks on the Materials Project's MatBench [51]. The scaled MAE is defined as the mean absolute error divided by the mean absolute deviation for the dataset and indicates how good a fit is similar to the R^2 coefficient. The best model on both benchmarks, ALIGNN [39], has a scaled MAE of 0.02 and 0.14 for energies of formation and bandgaps respectively. Furthermore, it is widely known that density-functional theory in the local-density-approximation (LDA) and the generalized-gradient-approximation (GGA) systematically underestimates band gaps [52][53][54]. This systematic error can be improved by the use of hybrid functionals [55], but an expansive database has yet to be created using this method. It is expected that the model that is fitted on the same data, carries the same bias. This should be kept in mind when discussing the use of GNNs trained on DFT data in high throughput searches, e.g. for large band gap materials.

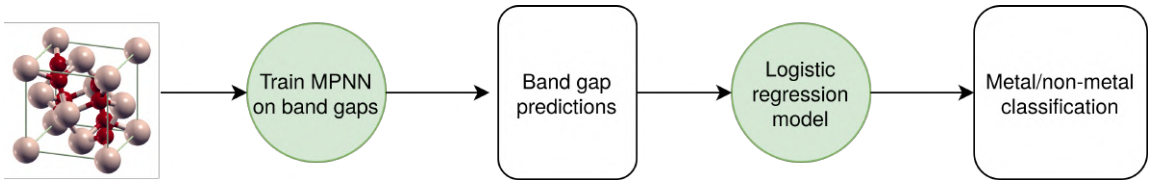

Figure 5.8: Workflow of band gap fitting using MPNN and post-classification using a logistic regression model. First, a message passing neural network is trained to predict band gaps from the crystal structure. These predictions are combined with the true band gap type label (as produced by the AFLOW calculation) to train a logistic regression model. This model produces a band gap cutoff threshold. Compounds with a predicted band gap below the cutoff are classified as metals, and compounds with predicted band gaps above the cutoff are predicted as non-metals. Following the procedure in [5], we first classify our test data as non-metal and metal and report classification metrics. We then report band gap regression metrics for the materials in the test dataset we classify as non-metal.

Before fitting the model, the dataset is cleaned by only using calculations with the PAW-PBE DFT-type label (similar to the selection when filtering enthalpy of formation in Section 5.3). The result of the band gap regression can be seen in Fig. 5.9. The high coefficient of determination of $R^2 = 0.930$ on the test set shows that despite the data distribution, the prediction of unseen data is relatively accurate. We also see that there is a strong dependence on the space group, when predicting the band gap (Fig. 5.7). The difference between cubic systems and other systems is even more evident than for the energy of formation model. However, the mean absolute error is underestimated, since the classes metals and non-metals are not balanced (about 80% of the structures in the AFLOW database are predicted by DFT to be metals). This leads to a bias in the model to make null predictions, i.e. predicting a band gap of 0, regardless of composition and structure. In order to have a less biased MAE, we calculate the MAE for non-metals only. To achieve this, the predictions for the band gap are used in a logistic regression, along with the true DFT band gap type to produce a model that classifies metals vs. non-metals/insulators. This workflow is illustrated in Fig. 5.8. In other publications this metal/non-metal classification is done

by training an additional model, using a similar architecture, with a classification loss, like binary cross-entropy [5][7][33].

We show that in contrast to common practice, a single model can be trained to estimate band gaps and classify metals/non-metals. The overall classification rate is high with a total accuracy of 0.984 (see Table 5.1), averaged over 10 training/validation/test splits. As the dataset is quite unbalanced with 80% metals, the total accuracy is not the best measure to be looked at. The area-under-curve of the receiver-operator characteristic gives a threshold independent metric that accounts for unbalanced classes and gives a more accurate representation of the classification, especially if the goal is to find non-metals or insulators. The AUC for the logistic regression model is relatively high with 0.98 (see Fig. 5.10). Other metrics to consider are sensitivity and specificity. If the correct classification of a metal is considered a true-positive result, the average sensitivity of the model is 0.994 and the average specificity is 0.934. This shows the expected bias towards predicting a structure to be a metal as a result of the unbalanced dataset classes. In other words, a non-metal is correctly classified 93.4% of the time (averaged over all models from using different training/validation/test splits).

We see that the model predicts one structure to have a band gap of -2 eV. This structure is SiO_2, in the hexagonal space group and trigonal-trapezoidal point group. This is a nonphysical prediction from our model which is not constrained to predict only positive bandgaps. It might be a result of the scaling and normalization of inputs, which is routine in all neural network models. More investigation is needed into how the output can be constrained to be positive. In Section 5.6, we show that oxides are predicted with lower accuracy than other species, therefore this outlier fits the observation with respect to space groups and species.

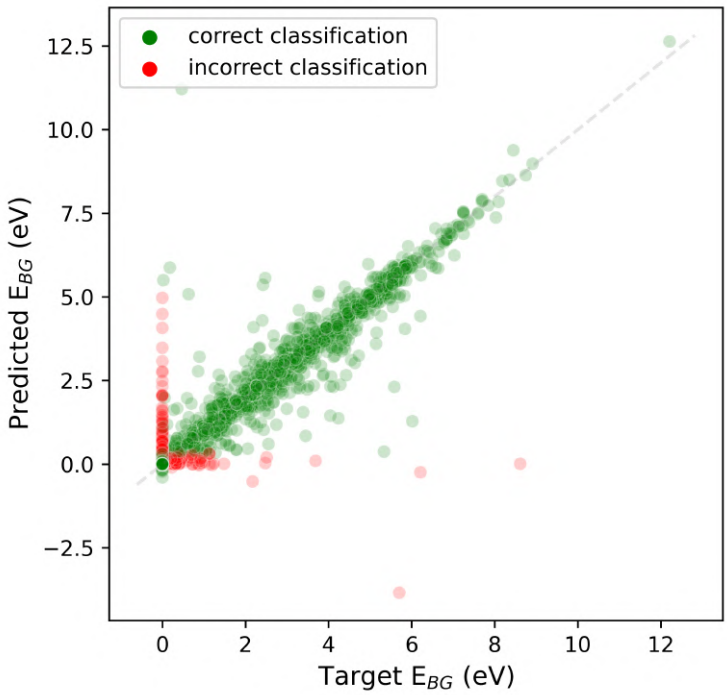

Figure 5.9: Band gap regression result on the test set overlaid with the binary classification in terms of metal vs. non-metal. The classifier calculates the threshold that best divides metals and non-metals using the band gap prediction from the GNN. This is done with a simple, binary logistic regression. One outlier with a negative predicted band gap can be seen, as there is no prior assumption made for the distribution of target band gaps. The outlier material is SiO_2.

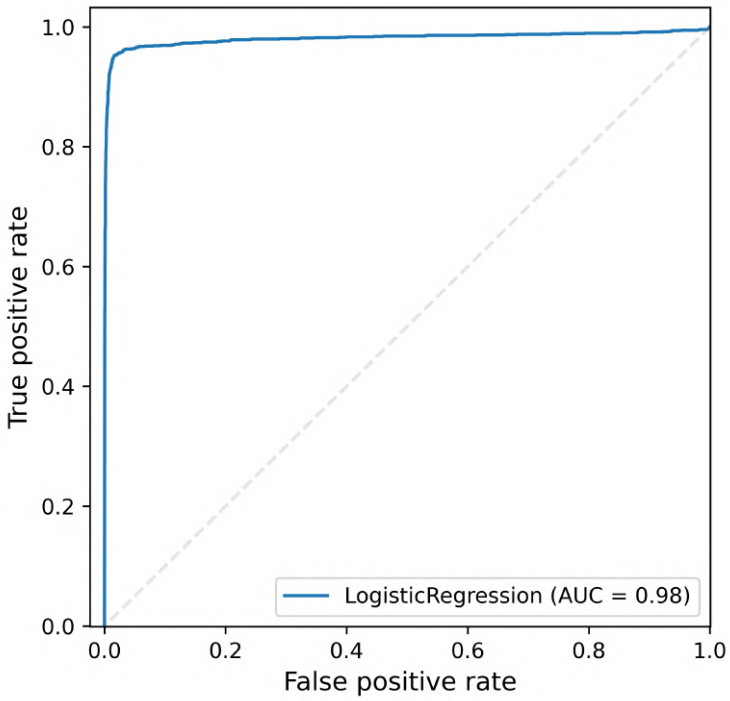

Figure 5.10: Band gap classification receiver-operator-characteristic (ROC) curve.

5.5 Cross validation and grid search

The hyperparameters of the base model in [31] were optimized for the OQMD and MP databases, thus the precision might not be optimal on a different dataset, e.g. AFLOW. For this reason, a full grid search of chosen hyperparameters is done in order to identify and easily visualize the important settings for the model. We opt to focus on optimizing the latent size, number of message passing steps, initial learning rate, and learning rate decay. The first two determine the model architecture and number of learnable parameters, while the two latter define the training process and parameter updates using the ADAM optimizer.

For each set of hyperparameters, the training is performed as described in Section 4.5, holding out the test split evaluation until the model with the best performance on the validation split is found. This ensures that the hyperparameter combination is not overfit on the specific test split and a truly separate split is evaluated on in order to assess the final predictive performance. The results of the grid search is shown in Fig. 5.11. The most important and also sensitive parameters are the latent size, number of message passing steps, and initial learning rate. Here, the best parameters are 128 latent nodes as hidden representation, five message passing steps and an initial learning rate of 1e-4 for the ADAM optimizer. We can see that compared to the hyperparameters in [31], the optimal model is deeper and not as wide when adapting to the AFLOW dataset. This also means that the model has fewer parameters, as the number of weights scales linearly with the number of message passing steps and quadratically with the latent size. The same can be said about time complexity of a gradient step, which scales linearly with the number of message passing steps and quadratically with the latent size.

However, this resulting optimal network configuration has quite high variance when repeating the training on different training splits and even with the same splits due to the stochastic nature of the training. Trends for model hyperparameters can be seen from the grid search, but if the true optimal model is desired, nested cross-validation [56] should be used. This produces an almost unbiased performance estimate with respect to the training-validation-test split. As the computational demand for the nested procedure is quite high, however, this evaluation is not in the scope of this thesis. In other publications, the prohibitive cost also deters researchers from using the nested cross-validation technique.

As a whole, the hyperparameters given in [31] are useful as base guesses for the optimal parameters for the AFLOW energy of formation fit, but more optimal parameters can be found using a simple grid search. These hyperparameters are used to fit the band gaps on the AFLOW structures, as there are more parameters with larger latent sizes and more message passing steps, which should be beneficial when fitting a more complicated, non-local function such as band gap, compared to simple energies, such as energy of formation per atom.

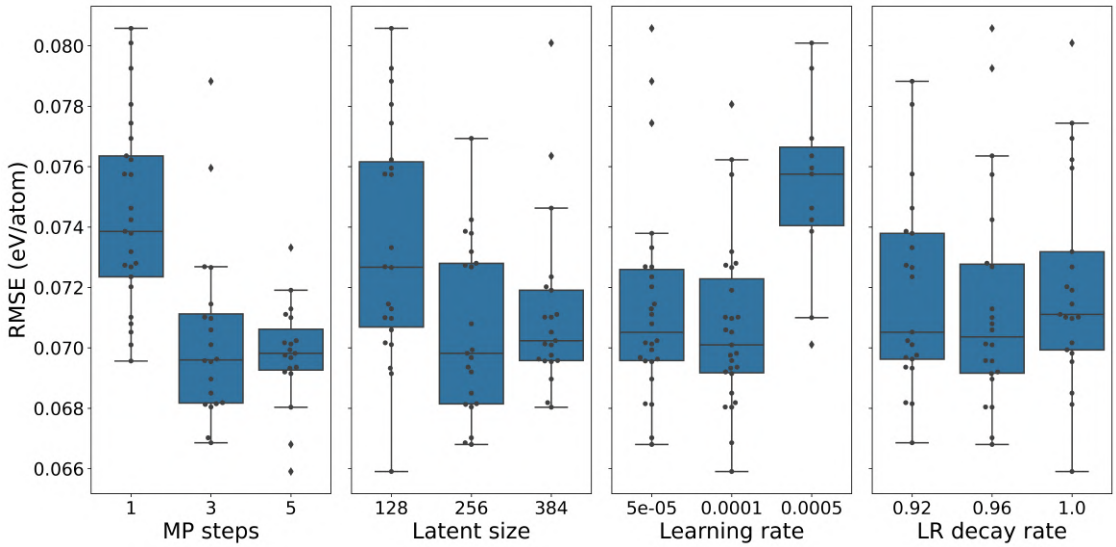

Figure 5.11: Grid search results for optimal hyperparameters, on the AFLOW dataset. Shown are the root mean squared errors (RMSE) on the validation split when fitting the energy of formation per atom, as the minimum of the learning curve, i.e. the lowest validation error evaluated during training. The model is evaluated every 50,000 gradient steps. For better visibility of the trends, some outlier points with non-converged models are not shown.

5.6 Species analysis

In order to better understand the models' capacity and shortcomings, we analyze its ability to predict the formation and band gap of unseen structures with varying atomic species. This shows in how far the prediction error depends on the number of examples per atomic species in the training set. It might also show that some classes of materials are harder to predict even though there are many training examples. In Fig. 5.12, it is shown that mostly the MAE decreases with increasing number of samples with the same species and there is an approximate upper bound on the MAE that decays roughly exponentially with the number of samples. We see that most samples are from the transition metal class of the Periodic Table, and the errors are clustered around an MAE of 0.01 eV/atom.

The biggest outliers are the elements neon and oxygen. The great predictive performance on neon compounds is easily explained by the fact that from the 26 compounds containing Neon in the dataset, all but one are elemental crystals with the same space group. This means that the structures are very similar and mostly describe different densities of the same material, making it easy for the model to interpolate. Oxides, i.e. compounds containing oxygen on the other hand seem to be hard to predict, despite there being more than 9000 oxides in the training split. One possible explanation for this is the complicated electronic structure of transition metal oxides which are a subgroup of rather strongly correlated materials that challenge the predictive accuracy of DFT. For this reason, often methods that go beyond DFT are used to correct for this behavior [57], such as the DFT+U correction in the AFLOW database [37]. This parameter is chosen on a per element basis in the calculation of the database entries, but it is not fed into the model. As a consequence the model has to infer the corrected formation energy without knowing if the calculation uses the DFT+U correction.

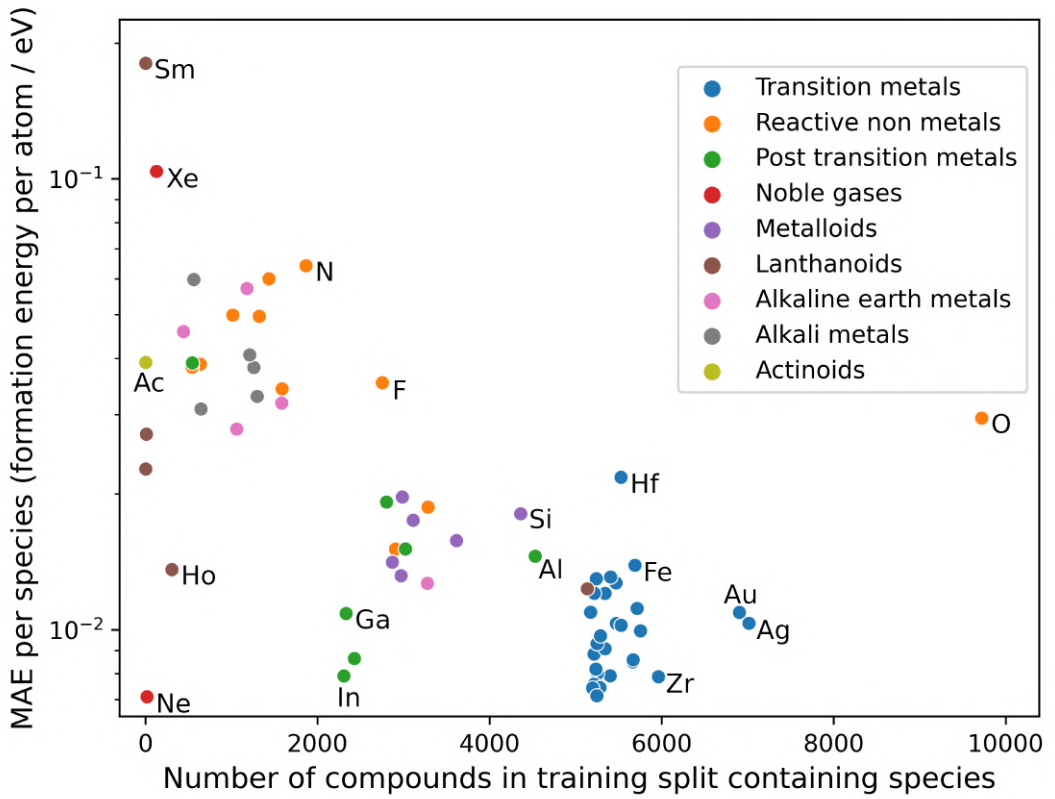

Figure 5.12: Prediction performance based on mean absolute error and number of species in the training split, when predicting formation energies on the AFLOW dataset. MAEs are based on the test data split. Some species are highlighted with a label, in order to show outliers or give examples for specific families in the Periodic Table.

The differences between prediction errors grouped with species become even more evident when predicting the band gap. In Fig. 5.13, the mean absolute error for band gap predictions depending on number of compounds with specific elements in the training split is shown. The MAEs of the tight cluster of transition metals are now spread from 0.001 eV to about 0.02 eV. This shows that the different numbers of valence electrons are more important when predicting band gaps, which makes sense, intuitively. We also see that now lanthanoids are predicted more precisely than noble gases, whereas it was the other way around for formation energies. Again, oxides are an outlier in this plot, and it can be hypothesized that this is again an effect of the more complicated interatomic interactions in oxides, and the deviation from other materials is more pronounced for band gaps.

This kind of analysis is important when applying the model on new domains, where no reference label is available and the validity of the prediction is important. If the application of this specific model is e.g. high-throughput search, then the most precise predictions should be expected for transition metals, but predictions for oxides are expected to be less accurate. If the goal is prediction on a specific material class, the training can also be adapted to be more precise on this specific class, by weighting the loss function more for the desired class. This kind of model specialization should be examined further, while also investigating the effect of smaller training data size, but is not in the scope of this thesis. The MAE has also been shown to drop off with increasing training data size [7][39][26].

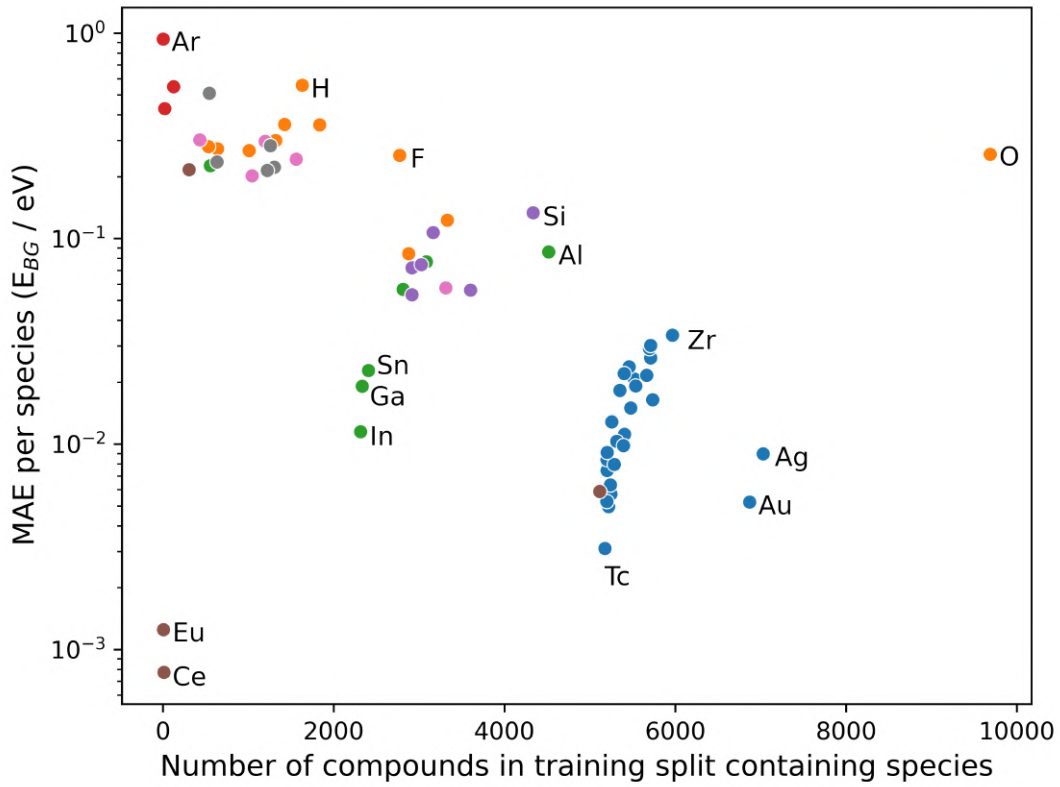

Figure 5.13: Prediction performance based on mean absolute error and number of species in the training split, when predicting band gaps on the AFLOW dataset. MAEs are based on the test data split. Color-code for the different element classes can be taken from Fig. 5.12. Some families are highlighted with a label, in order to show outliers or give examples for specific species.

5.7 Monte-Carlo-Dropout uncertainty

Reliable uncertainty estimates are important when deploying a machine learning model in a real application [58], as well as is in active learning, where data points are picked based on the uncertainty in a certain region of input space or in high throughput searches, when candidate structures to simulate or even synthesize are picked [59]. One possibility to get an uncertainty estimate is Monte-Carlo Dropout (MCD). The dropout enables stochastic predictions from a "virtual ensemble", and aggregating these predictions gives an uncertainty in the same way as a Gaussian Process would [60].

In order to add dropout into the model while not changing the model architecture too much, we opt to add additional funnelling layers after the aggregation of nodes in the readout function, containing dropout layers. These are tested with dropout rates of 0, 0.1, and 0.2. As can be seen in Fig. 5.14, the prediction standard deviation does not directly correlate very well with the absolute error of the prediction, for the training and test splits. A similarly absent correlation can be found in [59] for their experiment on the Online News Popularity dataset. In contrast, they achieved much better correlation (Pearson correlation coefficient equal to 0.93) on the CT (computed tomography) slices dataset [59].

Going back to results of MCD on AFLOW data, however, we can see better uncertainty estimates, when splitting the data into different subsets, with different numbers of training data. Looking at distributions of uncertainties when the data is split into crystal systems (Fig. 5.15), it is apparent that the model is more "certain" (i.e. lower standard deviation with MCD) for cubic systems, than all other symmetries. This confirms the expectation that the model is more certain for systems it has more training data on, since the dataset consists of about 87% cubic systems, but this effect is not reflected in the overall uncertainty depending on absolute error. The same

can be observed when dividing the test data into DFT+U corrected calculations and uncorrected calculations. In Fig. 5.16, we see that the model is more certain of predictions that are not corrected using the Hubbard U correction. This observation warrants further investigation, since the model might also be better trained on uncorrected calculations, due to the unbalanced dataset, with respect to number of corrected and uncorrected calculations.

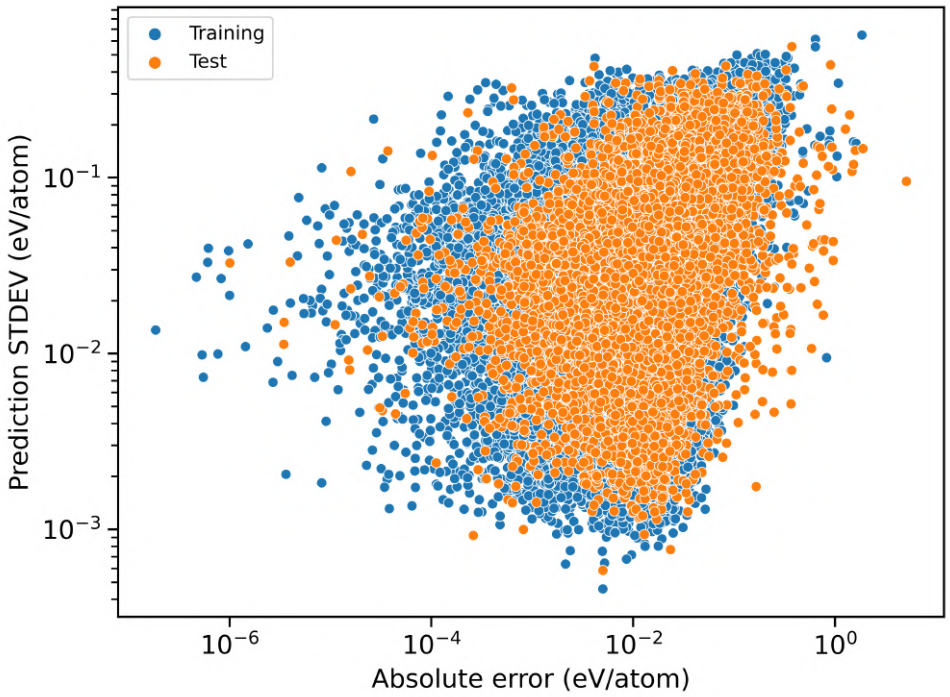

Figure 5.14: Monte-Carlo dropout uncertainties plotted against absolute errors for the different splits. For each structure from the AFLOW dataset, ten predictions of the energy of formation per atom are made using dropout with different seeds and the standard deviation is calculated.

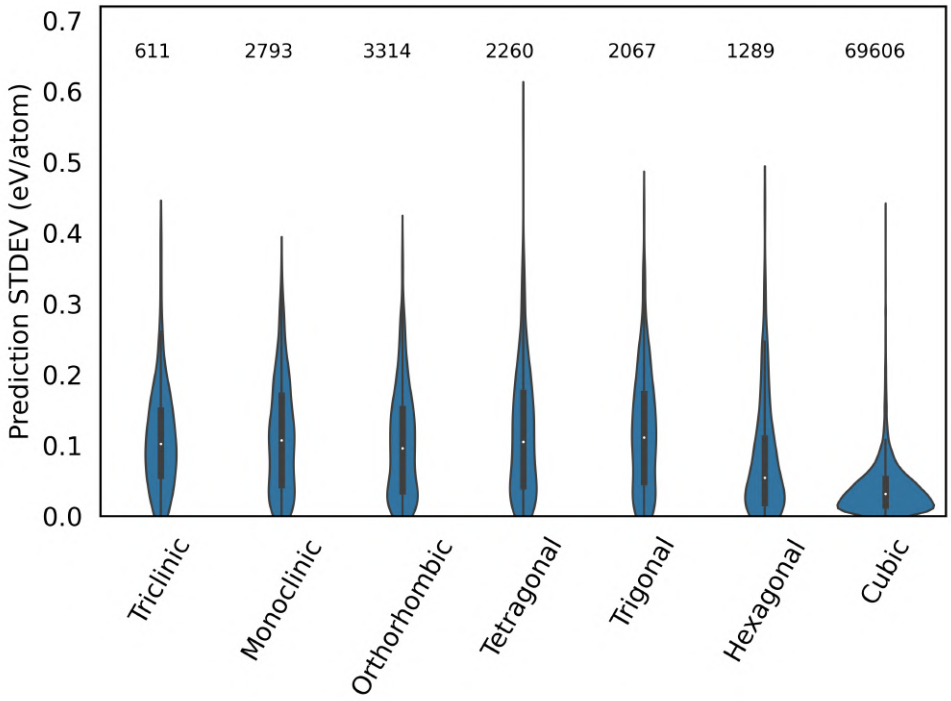

Figure 5.15: Violin plot of model uncertainties on test AFLOW dataset, targeting formation energies for different crystal systems. Violins show the distribution of standard deviations together with median and quarter-percentiles. Above each violin is the number of structures in the training split with the respective space group.

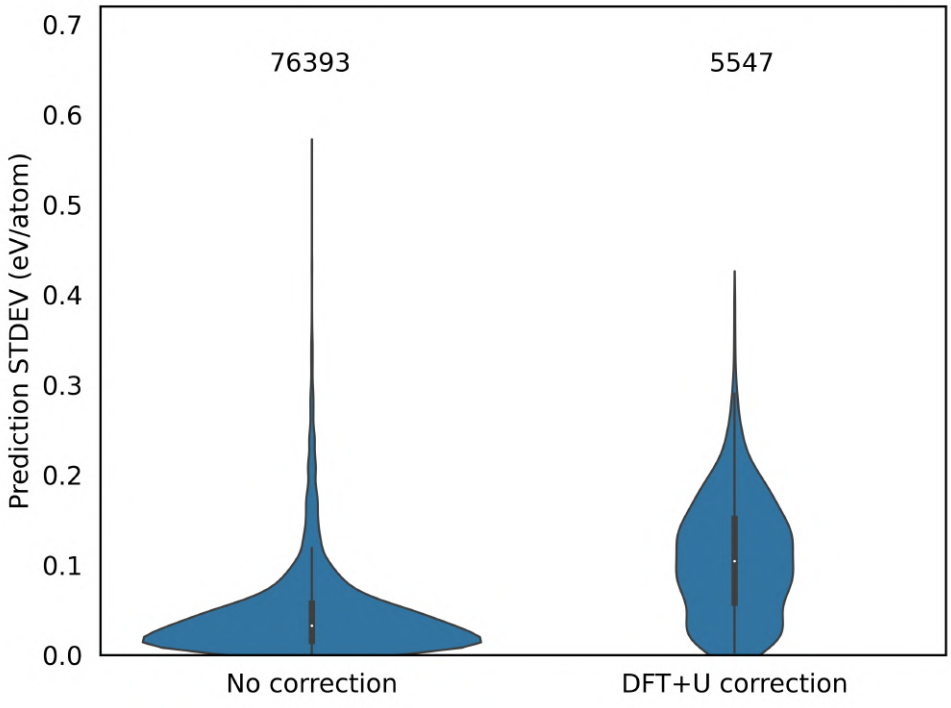

Figure 5.16: Distribution of uncertainties with Monte-Carlo dropout with ten samples per structure, for whether or not a Hubbard U correction was applied on the original structure. Predicted property is energy of formation per atom of the AFLOW database. All structures are taken from the test split. Also shown is the number of corrected and uncorrected calculations in the training split.

6. Conclusion and outlook

In this thesis, it was shown that the popular Message Passing framework for Graph Neural Networks can achieve high precision when predicting DFT calculated energetic and electronic properties. However, it is important to note that the model learns the same biases as the DFT calculations. This is even more important when looking at the results of band gap prediction, which carry a systematic error in DFT, compared to experiment. Here, the model predictions do not quite match DFT precision, after filtering out predicted metals, even though the classification between metal versus non-metal reaches high accuracy. All of these results are achieved on the AFLOW dataset, which has not been widely used as a machine learning benchmark, however the results are promising for future use of this data.

It is also shown in how far the data distribution affects the predictive power and uncertainties of the model. As expected, the model performs better and with more certainty in regions of the data where more training data exists. This is reflected when splitting the data in crystal classes, which shows the model's dependency on local topology when predicting formation energies, as this is a key feature of MPNNs. Furthermore, even information hidden to the model, like Hubbard U correction, which can only be inferred from pairwise distances in the relaxed structure, affects the model uncertainty in a significant way. In order to get a general uncertainty estimate, one that is linearly correlated to the actual absolute error, the model needs to be modified, potentially with dropout nodes in every hidden layer or with an ensemble of models. These experiments are left to future research, however.

There are some well known and proven limitations of MPNNs in which structures they can differentiate, even when the graphs are non-isomorphic [61]. This poses a challenge for theoreticians as the MPNNs limits are still relatively unexplored on real-world data, as there might be an interesting class of materials or chemicals that act very differently in the real world, but are indistinguishable for the model. A simple example of this is chirality, which the model cannot account for. Some of these problems are addressed with GNNs that take into account three-body terms and angles [25][39][26], but even these have representational limitations [61]. The challenges created by these limitations are still to be explored.

Applications that have been tested and those which have been hypothesized are plenty. The most obvious usage of a fast approximation of DFT energies and electronic properties is high-thoughput screening with a very large pool of possible structures, enabling very fine-grained filters for e.g. stability, mechanical, thermal and electronic properties. This would allow for the prototyping of very finely tuned materials that have the exact properties that are needed. This process can also be turned around with generative models, which are based on good predictive models like GNNs. With a generative model the high amount of initial structures can be reduced and structures with the desired properties are generated directly. The systematic errors of predictions based on DFT can also be reduced by fine tuning a model based on DFT data, using experimental data in a process called transfer learning.

The most important aspects when evaluating a machine learning model is not only the mean absolute error in regression or accuracy in prediction but also specific tests that probe the generalization capability of the model. While the models can be improved further and further in the performance on a specific test set of benchmark, there are still some questions that have to be kept in mind. First, do we need more accurate models? If the underlying training data is biased or if the application the predictions should be used for does not need more accurate predictions,

perhaps other aspects should be focused on. If the average absolute error is comparatively low, but the model is not able to generalize to some interesting materials or domains, then not overall performance should be targeted, but the domain of applicability should be questioned. Second, what are the valid domains in which the model has enough predictive power? With the material dataset used, good predictions can only be expected for relaxed structures, that are generated using the same methods as have been used for generating the training dataset. Structures that are not similar to the training dataset, i.e. are out of distribution, with respect to some input parameter or output target, are not expected to be well predicted by the model. Another aspect of this can be transferred from the computer vision field of machine learning. Here significant insights have been gained from transforming the input data, by masking it or adding small amounts of noise. This leads to insights in the attention to specific features in the model or how easy adversarial attacks are on the network. This kind of insight could also be useful in future research in order to make GNN models for materials science more robust and generalise better.

6. Bibliography

[1] Pierre Hohenberg and Walter Kohn. "Inhomogeneous electron gas". In: *Physical review* 136.3B (1964), B864.

[2] Stefano Curtarolo et al. "AFLOWLIB. ORG: A distributed materials properties repository from high-throughput ab initio calculations". In: *Computational Materials Science* 58 (2012), pp. 227–235.

[3] Anubhav Jain et al. "A high-throughput infrastructure for density functional theory calculations". In: *Computational Materials Science* 50.8 (2011), pp. 2295–2310.

[4] James E Saal et al. "Materials design and discovery with high-throughput density functional theory: the open quantum materials database (OQMD)". In: *Jom* 65.11 (2013), pp. 1501–1509.

[5] Olexandr Isayev et al. "Universal fragment descriptors for predicting properties of inorganic crystals". In: *Nature communications* 8.1 (2017), pp. 1–12.

[6] Keith T Butler et al. "Machine learning for molecular and materials science". In: *Nature* 559.7715 (2018), pp. 547–555.

[7] Tian Xie and Jeffrey C Grossman. "Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties". In: *Physical review letters* 120.14 (2018), p. 145301.

[8] Shaoqing Ren et al. "Deep residual learning for image recognition". In: *CVPR*. Vol. 2. 2016, p. 4.

[9] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[10] Kathryn Tunyasuvunakool et al. "Highly accurate protein structure prediction for the human proteome". In: *Nature* 596.7873 (2021), pp. 590–596.

[11] Justin Gilmer et al. "Neural message passing for quantum chemistry". In: *International conference on machine learning*. PMLR. 2017, pp. 1263–1272.

[12] Kristof Schütt et al. "Schnet: A continuous-filter convolutional neural network for modeling quantum interactions". In: *Advances in neural information processing systems* 30 (2017).

[13] Claudia Draxl and Matthias Scheffler. "NOMAD: The FAIR concept for big data-driven materials science". In: *Mrs Bulletin* 43.9 (2018), pp. 676–682.

[14] Tian Xie et al. "Atomistic graph networks for experimental materials property prediction". In: *arXiv preprint arXiv:2103.13795* (2021).

[15] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[16] Dipendra Jha et al. "Elemnet: Deep learning the chemistry of materials from only elemental composition". In: *Scientific reports* 8.1 (2018), pp. 1–13.

[17] Lauri Himanen et al. "DScribe: Library of descriptors for machine learning in materials science". In: *Computer Physics Communications* 247 (2020), p. 106949.

[18] Antonia Creswell et al. "Generative adversarial networks: An overview". In: *IEEE Signal Processing Magazine* 35.1 (2018), pp. 53–65.

[19] Yoshua Bengio, Aaron Courville, and Pascal Vincent. "Representation learning: A review and new perspectives". In: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), pp. 1798–1828.

[20] Thomas N Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907* (2016).

[21] Jonathan Godwin et al. "Simple gnn regularisation for 3d molecular property prediction and beyond". In: *International conference on learning representations*. 2021.

[22] Michael A Nielsen. *Neural networks and deep learning*. Vol. 25. Determination press San Francisco, CA, USA, 2015.

[23] Marco Gori, Gabriele Monfardini, and Franco Scarselli. "A new model for learning in graph domains". In: *Proceedings. 2005 IEEE international joint conference on neural networks*. Vol. 2. 2005. 2005, pp. 729–734.

[24] Gabriele Corso et al. "Principal neighbourhood aggregation for graph nets". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 13260–13271.

[25] Johannes Klicpera, Janek Groß, and Stephan Günnemann. "Directional message passing for molecular graphs". In: *arXiv preprint arXiv:2003.03123* (2020).

[26] Simon Batzner et al. "E (3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials". In: *Nature Communications* 13.1 (2022), pp. 1–11.

[27] Peter W Battaglia et al. "Relational inductive biases, deep learning, and graph networks". In: *arXiv preprint arXiv:1806.01261* (2018).

[28] Patrice Y Simard, David Steinkraus, John C Platt, et al. "Best practices for convolutional neural networks applied to visual document analysis." In: *Icdar*. Vol. 3. 2003. Edinburgh. 2003.

[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012).

[30] Raghunathan Ramakrishnan et al. "Quantum chemistry structures and properties of 134 kilo molecules". In: *Scientific data* 1.1 (2014), pp. 1–7.

[31] Peter Bjørn Jørgensen, Karsten Wedel Jacobsen, and Mikkel N Schmidt. "Neural message passing with edge updates for predicting properties of molecules and materials". In: *arXiv preprint arXiv:1806.03146* (2018).

[32] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[33] Chi Chen et al. "Graph networks as a universal machine learning framework for molecules and crystals". In: *Chemistry of Materials* 31.9 (2019), pp. 3564–3572.

[34] Anubhav Jain et al. "Commentary: The Materials Project: A materials genome approach to accelerating materials innovation". In: *APL materials* 1.1 (2013), p. 011002.

[35] Georg Kresse and Jürgen Furthmüller. "Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set". In: *Physical review B* 54.16 (1996), p. 11169.

[36] Vinay I Hegde et al. "Reproducibility in high-throughput density functional theory: a comparison of AFLOW, Materials Project, and OQMD". In: *arXiv preprint arXiv:2007.01988* (2020).

[37] Camilo E Calderon et al. "The AFLOW standard for high-throughput materials science calculations". In: *Computational Materials Science* 108 (2015), pp. 233–238.

[38] Frisco Rose et al. "AFLUX: The LUX materials search API for the AFLOW data repositories". In: *Computational Materials Science* 137 (2017), pp. 362–370.

[39] Kamal Choudhary and Brian DeCost. "Atomistic Line Graph Neural Network for improved materials property predictions". In: *npj Computational Materials* 7.1 (2021), pp. 1–8.

[40] John P Perdew, Kieron Burke, and Matthias Ernzerhof. "Generalized gradient approximation made simple". In: *Physical review letters* 77.18 (1996), p. 3865.

[41] Tim Bechtel. *jraph_MPEU*. https://github.com/tisabe/jraph_MPEU. 2022.

[42] Rudolf Gross et al. "Festkörperphysik". In: *Festkörperphysik*. de Gruyter, 2018.

[43] James Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.2.5. 2018. URL: `http://github.com/google/jax`.

[44] Tom Hennigan et al. *Haiku: Sonnet for JAX*. Version 0.0.3. 2020. URL: `http://github.com/deepmind/dm-haiku`.

[45] Matteo Hessel et al. *Optax: composable gradient transformation and optimisation, in JAX!* Version 0.0.1. 2020. URL: `http://github.com/deepmind/optax`.

[46] Jonathan Godwin* et al. *Jraph: A library for graph neural networks in jax.* Version 0.0.1.dev. 2020. URL: `http://github.com/deepmind/jraph`.

[47] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[48] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016).

[49] Anubhav Jain et al. "The Materials Project: A materials genome approach to accelerating materials innovation". In: *APL Materials* 1.1 (2013), p. 011002. ISSN: 2166532X. DOI: `10.1063/1.4812323`. URL: `http://link.aip.org/link/AMPADS/v1/i1/p011002/s1%5C&Agg=doi`.

[50] Scott Kirklin et al. "The Open Quantum Materials Database (OQMD): assessing the accuracy of DFT formation energies". In: *npj Computational Materials* 1.1 (2015), pp. 1–15.

[51] Alexander Dunn et al. "Benchmarking materials property prediction methods: the Matbench test set and Automatminer reference algorithm". In: *npj Computational Materials* 6.1 (2020), pp. 1–10.

[52] John P Perdew. "Density functional theory and the band gap problem". In: *International Journal of Quantum Chemistry* 28.S19 (1985), pp. 497–523.

[53] Diola Bagayoko. "Understanding density functional theory (DFT) and completing it in practice". In: *AIP Advances* 4.12 (2014), p. 127104.

[54] Lu J Sham and Michael Schlüter. "Density-functional theory of the energy gap". In: *Physical review letters* 51.20 (1983), p. 1888.

[55] Hai Xiao, Jamil Tahir-Kheli, and William A Goddard III. "Accurate band gaps for semiconductors from density functional theory". In: *The Journal of Physical Chemistry Letters* 2.3 (2011), pp. 212–217.

[56] Sebastian Raschka. "Model evaluation, model selection, and algorithm selection in machine learning". In: *arXiv preprint arXiv:1811.12808* (2018).

[57] Subhasish Mandal et al. "Systematic beyond-DFT study of binary transition metal oxides". In: *npj Computational Materials* 5.1 (2019), pp. 1–8.

[58] Jonas Busk et al. "Calibrated uncertainty for molecular property prediction using ensembles of message passing neural networks". In: *Machine Learning: Science and Technology* 3.1 (2021), p. 015012.

[59] Evgenii Tsymbalov, Maxim Panov, and Alexander Shapeev. "Dropout-based active learning for regression". In: *International conference on analysis of images, social networks and texts*. Springer. 2018, pp. 247–258.

[60] Yarin Gal and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning". In: *international conference on machine learning*. PMLR. 2016, pp. 1050–1059.

[61] Vikas Garg, Stefanie Jegelka, and Tommi Jaakkola. "Generalization and representational limits of graph neural networks". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 3419–3430.

A. Selbstständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und noch nicht für andere Prüfungen eingereicht habe. Sämtliche Quellen einschließlich Internetquellen, die unverändert oder abgewandelt wiedergegeben werden, insbesondere Quellen für Texte, Grafiken, Tabellen und Bilder, sind als solche kenntlich gemacht. Mir ist bekannt, dass bei Verstößen gegen diese Grundsätze ein Verfahren wegen Täuschungsversuchs bzw. Täuschung eingeleitet wird.

Berlin, 26.10.2022 Tim Bechtel