Machine Learning Model for the Atomic Structure Determination of Molecules in Atomic Force Microscopy Images

Abstract

This project aims to improve the accuracy of an Atomic Structure Determination (ASD) program for the analysis of Atomic Force Microscopy (AFM) images of molecules. The ASD model is based on a Machine Learning (ML) framework that identifies the locations and species of the atoms in the molecule and also predicts the bonds between them (i.e. it reconstructs the molecular graph).
The objective of the project is pursued through the integration of a new feature to characterise the atoms in the molecule: the so-called SOAP (Smooth Overlap of Atomic Positions) molecular descriptor.

1. Introduction

Scanning Probe Microscopy (SPM) is an umbrella term that encompasses various experimental techniques for the study of the surfaces of materials at the nanoscale level. The techniques that fall under the SPM category all have in common the following facts:

- the active part of the measuring device is a nano-sized probe of some kind;
- the probe interacts strongly with the imaged material almost at an atomistic level;
- the probe provides local, almost pin-point precision, information about the microscopic structure of the material;
- the probe collects multi-channel information about the structure of the material;
- the probe is moved along the surface of the material according to a raster pattern in order to generate an image of the surface.

Within this group the most common techniques are Scanning Tunnelling Microscopy (STM) and Atomic Force Microscopy (AFM). The former can be used on conductors only, the latter on insulators as well. Given the focus of this project on the analysis of AFM images, a detailed description of this technique is now provided.

In AFM the experimental probe consists of a nano-sized silicon or silicon nitride tip attached to an oscillating cantilever. The tip is mechanically brought in close proximity to the sample and the effects caused by the tip-sample interactions are monitored using optical equipment. Without going into much detail, it is possible to measure the torsion of the cantilever and the change in amplitude, frequency and phase of the oscillations. The latter ones are particularly useful to extract the force experienced by the tip, which in turn can be used to control the separation of the tip from the sample surface.
Usually the tip-sample separation is regulated by a feedback mechanism that ensures that the force on the tip remains at a constant level. The output of the AFM scan can then be expressed either as a tip height VS xy position or as a frequency (or amplitude) VS xy position.
In recent years, AFM has reached sub-Armstrong resolution so that it has become possible to image molecules on surfaces.

Although the optimisation of the tip and scan parameters allows to obtain very high resolution AFM images, their analysis can still be a daunting task even for an experienced AFM user. For example, interpretation becomes particularly difficult when a molecule extends outside of the plane of the surface. In light of this, the development of analysis tools for the study of AFM images of molecules is invaluable for advancing the applicability of AFM in this area.

In this context, Niko Oinonen developed a Machine Learning (ML) framework for the Atomic Structure Determination (ASD) of molecules in AFM images. This ML model firstly identifies the location of the atoms in the image and secondly iteratively constructs the graph of the molecules by classifying the identified atoms and tracing the edge connections (chemical bonds) between them.
An example of the model output for some AFM image inputs is given in **fig. 1**. As it can be seen, the performance of the model is very good for small planar molecules (**A**), while it deteriorates when the molecules emerge from the plane of the surface (**B**) or when the molecules are very large (**C**). The model was mainly trained on AFM images of molecules on a grid of size 0-18Å, which results in an evident performance drop when the atoms are located outside of this region. This causes the very inaccurate prediction on the right hand side of the molecule in **fig. 1** (**C**).

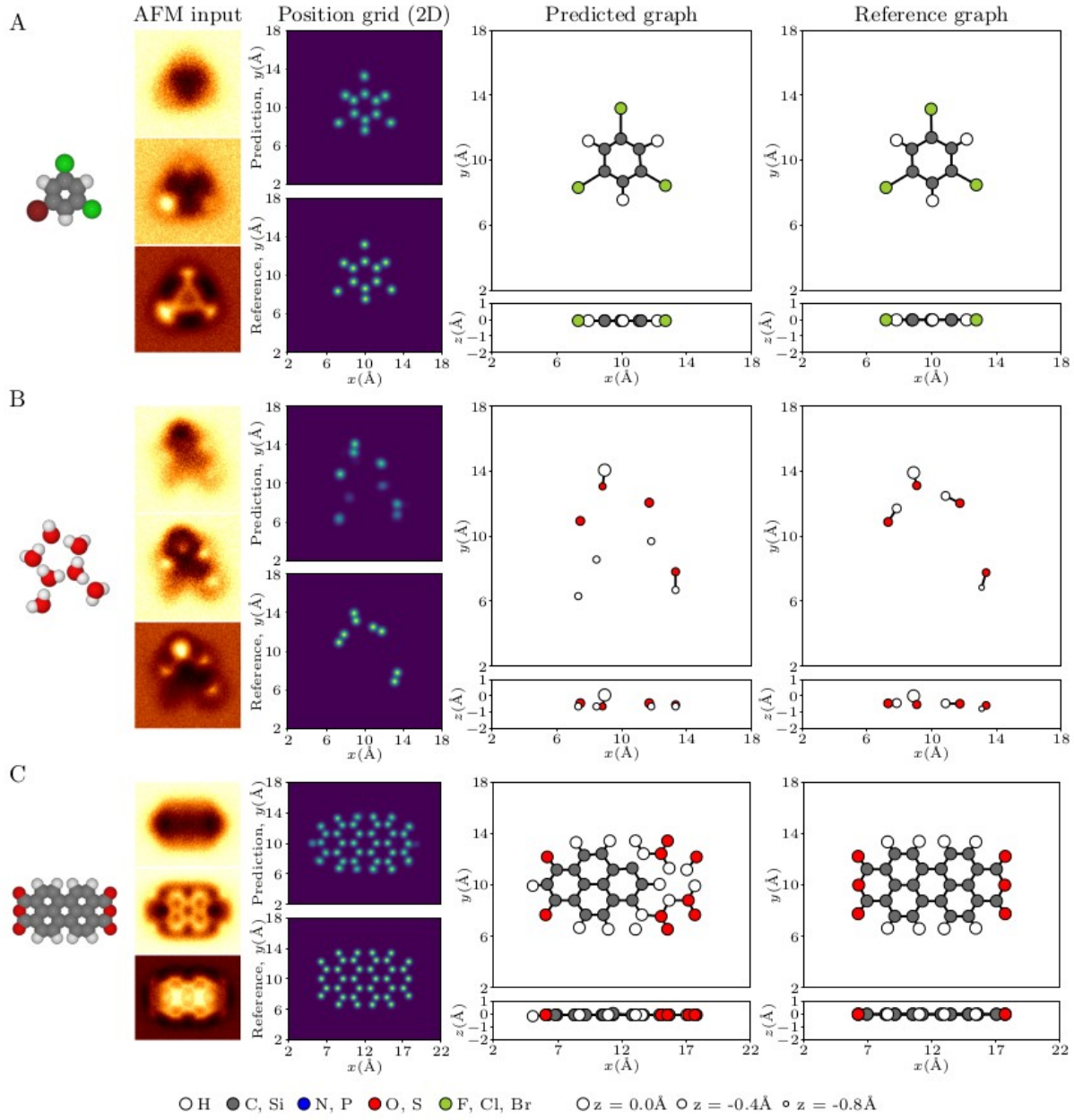

Figure 1: Model outputs for: (A) 1-bromo-3,5-dichlorobenzene, (B) a cluster of water molecules, (C) perylenetetracarboxylic dianhydride.

This behaviour of the model is more clearly manifested in **fig. 2** which shows the model predictions when the molecules in **fig. 1** are translated 2Å to the left. Not only the edge prediction worsens, but also the model capacity to classify the atoms sensibly decreases.
One of the most evident reasons for this behaviour of the model is the fact that the prediction mechanism strongly depends on the absolute coordinates of the atoms, so that if these are shifted beyond the region of training the model effectively has not accumulated enough information to

perform the classification and/or edge prediction task. In addition to that, a strong dependence on absolute coordinates results in the model having to learn rotational and translational invariance from the data, which is not desirable since this explosively increases the amount of data required for training.

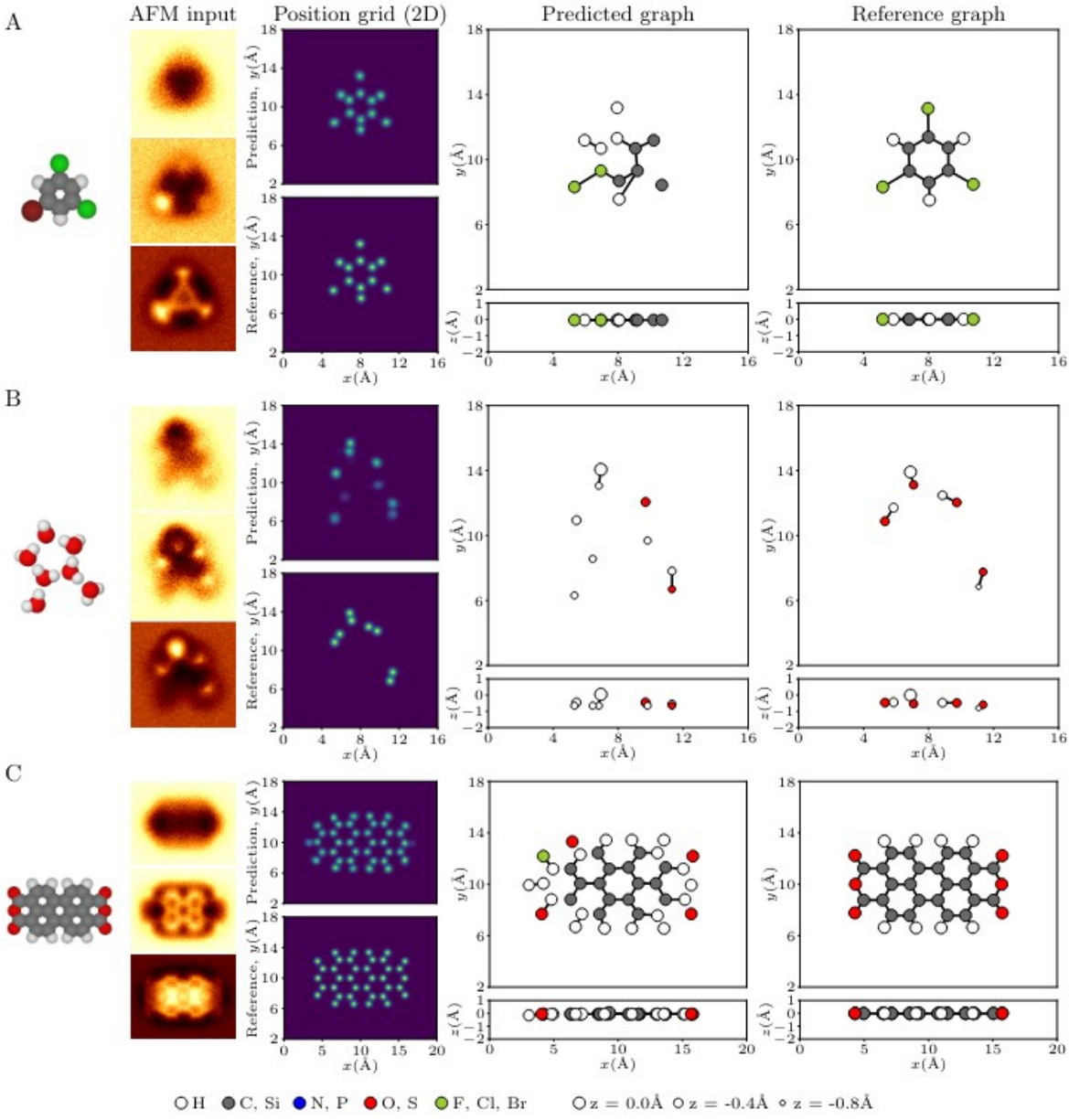

Figure 2: Model outputs for the same molecules in fig. 1 when the images are translated 2Å to the left.

Given these issues, the project I worked on from June 17 2022 to August 31 2022 explored the possibility of addressing the weaknesses of the current model by introducing an engineered feature into the ML system: the Smooth Overlap of Atomic Positions (SOAP) descriptor.
This is a man-made mathematical object that describes the geometry of molecular graphs. Each atom in the molecule has its own SOAP descriptor – that is SOAP is a local feature – and such a descriptor is also rotationally and translationally invariant. These characteristics make it a suitable candidate for integration into the ASD model.

2. What is SOAP?

Molecular descriptors are a class of mathematical entities used in Chemistry and Material Science to characterise the geometry of molecular structures and to quantify the similarity or discrepancy

between them. Within the context of ML, these mathematical descriptors found wide applicability as inputs for NN in the context of property prediction or structure discovery of new molecules.
In particular, the **SOAP (Smooth Overlap of Atomic Positions) descriptor** has proven to be a very elegant and robust strategy to describe coordination (geometrical) environments in a way that is naturally invariant with respect to translations, rotations and permutations of atoms [10] [12].
The definition of the SOAP descriptor for an atom in a molecule starts from the representation of the local density of atoms in the vicinity of the central atom as a sum of Gaussian functions with variance σ^2 centred on each of the neighbours of the central atom, as well as on the central atoms itself [12]:

$$\rho_\chi(\vec{r}) = \sum_{i \in \chi} \exp\left(-\frac{\left|\vec{x}_i - \vec{r}\right|^2}{2\sigma^2}\right) \tag{1}$$

where χ indicates the chemical environment with respect to the central atom, that is the coordinate distribution of the other atoms taking that atom as origin of the coordinate axes.
Given two chemical environments, χ and χ', the SOAP similarity kernel is defined as the overlap of the two local atomic neighbour densities, integrated over all three-dimensional rotations $\hat{R}$ [12], in symbols:

$$\tilde{k}(\chi, \chi') = \int d\hat{R} \left| \int \rho_\chi(\vec{r}) \rho_{\chi'}(\hat{R}\vec{r})\, d\vec{r} \right|^2 \tag{2}$$

This integral can be evaluated analytically by expanding the atomic densities in terms of tesseral spherical harmonics and a radial basis set of functions (usually of the form $g(r) = r^n e^{-r^2/\sigma^2}$):

$$\rho_\chi(\vec{r}) = \sum_{blm} c_{blm}\, g_b(r)\, Y_{lm}(\theta, \phi) \tag{3}$$

which in turn can be used to define the SOAP power vector:

$$p(\chi)_{b_1 b_2 l} = \pi \sqrt{\frac{8}{2l+1}} \sum_m \left(c_{b_1 l m}\right)^* c_{b_2 l m} \tag{4}$$

through which the similarity kernel can be expressed as:

$$\tilde{k}(\chi, \chi') = \vec{p}(\chi) \cdot \vec{p}(\chi') \tag{5}$$

Equations (4) and (5) can be obtained by exploiting the SO(3) group symmetry properties of the tesseral spherical harmonics expressed by the Wigner rotation matrices.
To be more precise, (4) is the SOAP power vector for single species molecules, when more than a chemical species is present, it is not complicated to show that the SOAP power vector becomes:

$$p(\chi)^{\alpha\beta}_{b_1 b_2 l} = \pi \sqrt{\frac{8}{2l+1}} \sum_m \left(c^\alpha_{b_1 l m}\right)^* c^\beta_{b_2 l m} \tag{6}$$

where α and β denote the two chemical species.

The SOAP power vector for a given molecule can be efficiently computed using the SOAP object of the **DScribe** library of molecular descriptors [10]. This library provides a limited selection of radial basis functions from which the spherical Gaussian type radial basis, $g_{nl}(r) = \sum_{n'=1}^{n_{max}} \beta_{nn'l} r^l e^{-\alpha r^2}$, was chosen for this project.

When defining the SOAP object, the following hyper-parameters must be specified:
- l_{max}: the maximum l value for the spherical harmonics (runs from 0 to l_{max});
- n_{max}: the maximum n value for the spherical Gaussian functions (runs from 1 to n_{max});
- r_{cut}: cut-off local region in Armstrong for the smearing of the atomic positions;
- σ/α: the standard deviation of the Gaussian distributions used in the smearing of the atomic positions and as basis functions;
- the chemical species present in the molecules.

In the ASD model SOAP was instantiated with the parameter settings: $l_{max} = 2$, $n_{max} = 3$, $r_{cut} = 4$, $\sigma/\alpha = 1$ and number of different chemical classes = 5. Therefore, the SOAP power vector for a chemical environment *Z-Z'* (where *Z* is the lighter chemical species/class) contained a total of 27 entries. It

should be highlighted that for *Z-Z* environments (environments involving the same chemical species/class) the power vector output had size 18 since the symmetric components were not calculated. Hence, this type of output had to be reshaped to cast it to the same size and shape of the other inter-species SOAP power vectors.

The ASD model classifies the atoms into 5 different groups (see the bottom of **fig. 1** and **2**), so that the number of different *Z-Z'* combinations (i.e. different species-specific chemical environments) totals to 15. In other words, for each atom in the molecular graph, SOAP provides 15 different vector features of size 27.
In its initial formulation, the ASD model characterises the atoms in the molecular graph just by their coordinate location and one-hot class. Thus, the integration of the SOAP power vector as an additional atomic feature is a significant improvements in terms of local, rotationally and translationally invariant, geometrical information available to the ML system.

3. The original model and relevant theory

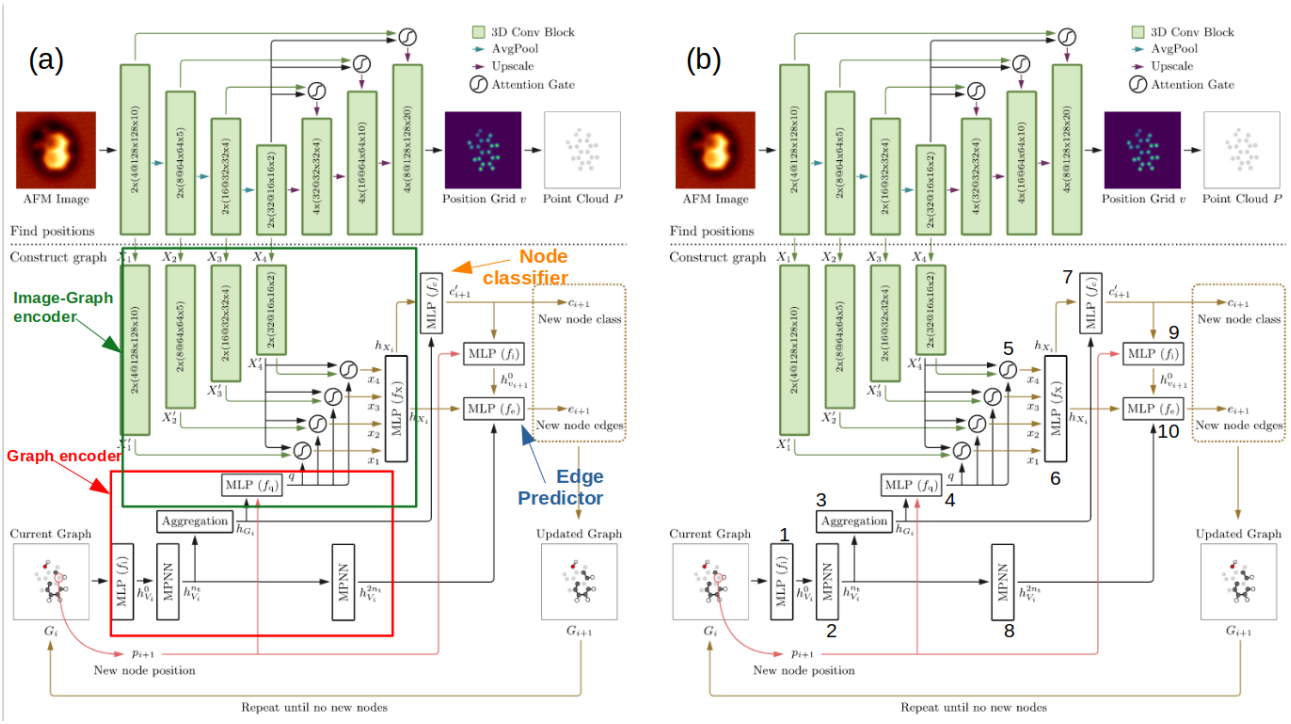

Figure 3: (a) Processing blocks in ASD model, (b) ASD model workflow.

Fig. 3 (**a**) shows a diagram of the ASD model prior to any modification.
The model divides up into two main parts:

- a **Convolutional Neural Network** (**CNN**) block: this part of the system processes a 10-level AFM image and outputs a 20 level position grid where each atom in the grid is represented by a normal distribution centred about the atom location. The position grid is further processed by a peak finding algorithm (based on a template matching method) that extracts the coordinates of all the identified atoms;
- a classifier block: this part of the network takes the atom coordinates along with the information from the first 4 CNN layers, and iteratively builds up the molecular graph by classifying each atom (referred as a node in this context) and by predicting the edge connections. The information processing mechanism of this part of the model strongly depends on a type of network known as **Graph Neural Network** (**GNN**) which has become increasingly popular in recent years for the construction of ML models whose data structures are easily represented by graphs (e.g. molecules).

It is not possible, and not even necessary, to provide here a detailed description of the entirety of the model. What follows is then a more detailed break-down of the graph construction part of the network. The reader interested in fully grasping the model workflow is invited to consult Niko Oinonen's original publication [1].

Starting from the "Current Graph" on the bottom left part of the diagram (**fig. 3** (**b**)), the graph constructor firstly initialises the node features (1), which comprise the coordinates and the one-hot class of the node. Then, the node hidden state vectors are passed to the GNN, which in this case is a **Message Passing Neural Network** (**MPNN**) (2).

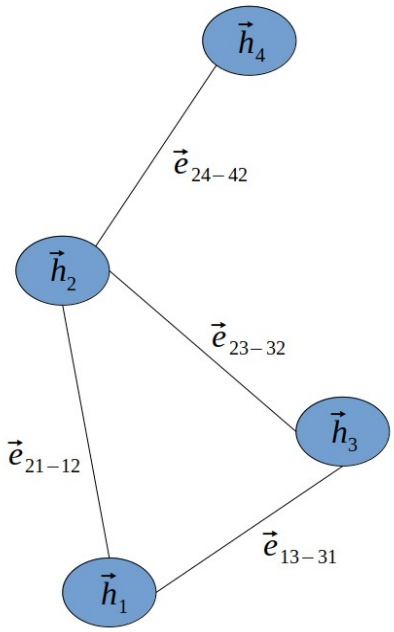

Figure 4: Message Passing Graph Neural Network diagram

Referring to **fig. 4**, which displays a graph with non-directional edge connections, this network architecture updates the node hidden states by propagating messages between the nodes connected by edges. The MPNN algorithm for undirected edge connections proceeds as follows:

1. a message vector, *e*, is calculated for each edge using the node hidden states of the connected nodes;
2. the non-directionality of the edge connections is enforced by symmetrizing the message computation: the message is calculated for both node orderings and the two outputs are averaged;
3. the node hidden states are updated by combining the initial hidden states with the messages of the edges arriving at those nodes;
4. multiple message passing iterations are performed in order to increase the perceptive field of a given node.

In the initial ASD model, the node hidden states and the messages had size 20, and the message aggregation was performed using a **Gated Recurrent Unit** (**GRU**), one of the most common types of **Recurrent Neural Network** (**RNN**).

After 3 rounds of message passing, the node hidden states are aggregated into a graph encoding vector of size 128, h_{G_i} (3). This vector, along with the coordinates of the new node to be classified (which is combined to h_{G_i} using a **Multi Layered Perceptron** (**MLP**) (4)), are then combined with the AFM image CNN encoder channels using an attention gate mechanism (more about attention in the next section). This mixing of image encoding, graph encoding and new node location is occurring in the part of the graph constructor model that mirrors the first 4 CNN layers (5). The outputs of the attention gates (x_1, x_2, x_3, x_4) are then aggregated using a MLP to generate an image-graph encoding, h_{X_i} (6). This is the input of a node classifier MLP that outputs the class of the new node (7).

The edge prediction task requires further processing of the graph. The outputs of the first round of message passing undergo further MPNN iterations to produce the final node hidden states (8). Moreover, the new node location and the newly predicted class are used to initialise a node hidden state for the new node (9). The message passed hidden state vectors belonging to the original graph and the new node hidden state are then fed to an edge predictor MLP that performs a binary classification task between all possible new-node/old-node pairs to decide which nodes in the original graph are connected to the new node (10).
After this has been completed, the updated graph is fed back to the graph constructor to continue the ASD process.

4. Proposed model modifications and relevant theory

Fig. 5 and **6** show how I intend to modify the original ASD model. The guiding principle undelying these changes is the integration of the SOAP power vector into the system as an additional node feature.

This is accomplished in two parallel ways:

1. the use of the SOAP vector as an extra node feature in the node hidden states initialisation;
2. the prediction of the SOAP vector for the new node using a SOAP predictor block.

Point 1 is useful to add a geometrical, node-specific and equivariant feature to the hidden states definition of the GNN. This intuitively should provide extra information about the system geometry in the ASD model.

Point 2 is a direct use of SOAP as a learning target with the purpose of explicitly forcing the model to learn about the geometry of recurring chemical environments and to learn about rotational and translational invariance. In fact, on one hand molecules typically present chemical groups with a well defined geometrical structure, while, on the other, SOAP is a local equivariant descriptor.

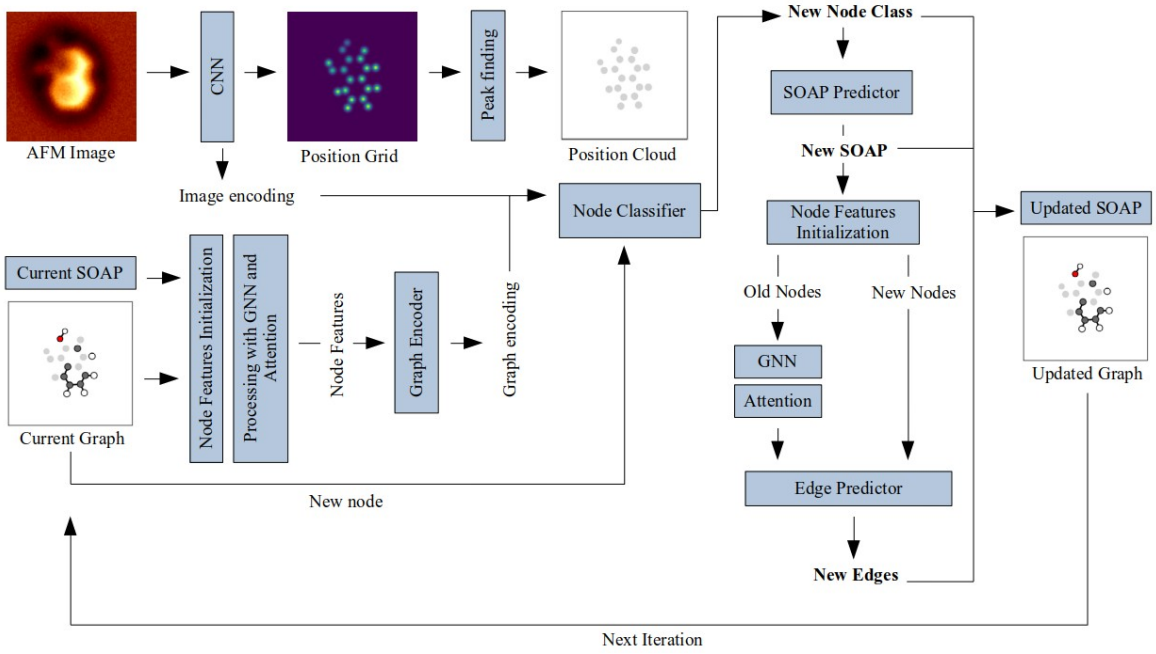

Figure 5: The ASD model

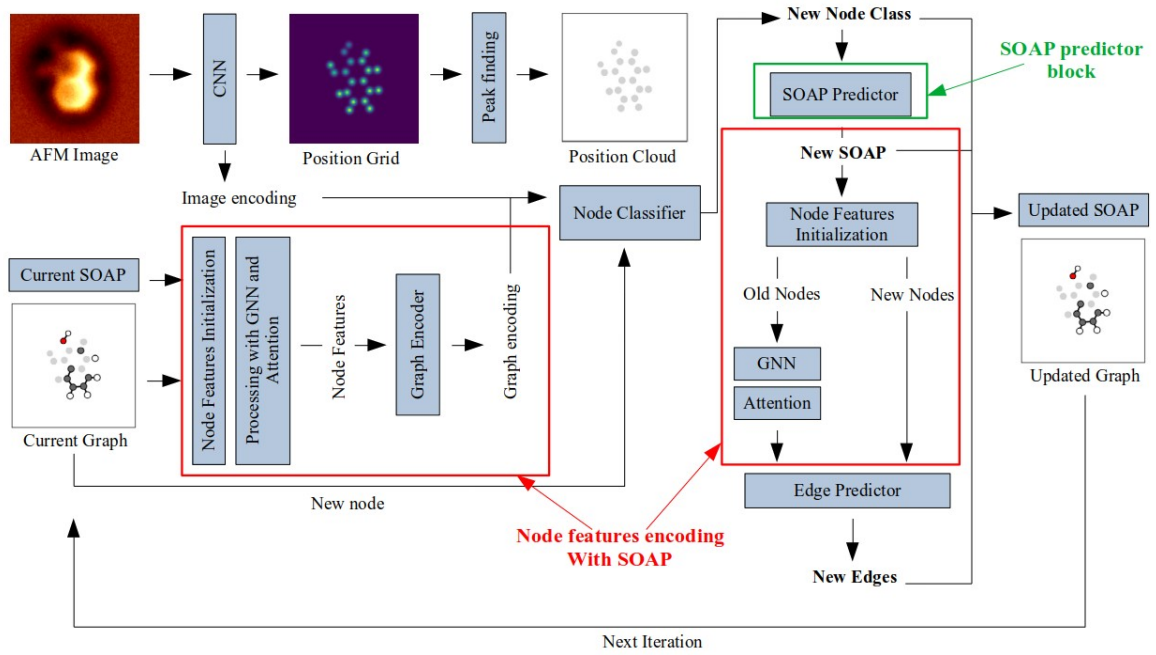

Figure 6: The SOAP vectors are added to the characterisation of the node features and a SOAP predictor block is included in the model.

In the following sections an in-depth explanation of the SOAP-based node features encoding and of the SOAP predictor block will be provided. However, before that it is important to introduce the so-called **attention layer** (or mechanism) to the ML toolkit that will be utilised.

The attention mechanism was introduced for the first time in the 2017 article "Attention is all you need" by Ashish Vaswani et al. [4]. The architecture of this NN layer is displayed in **fig. 7**.

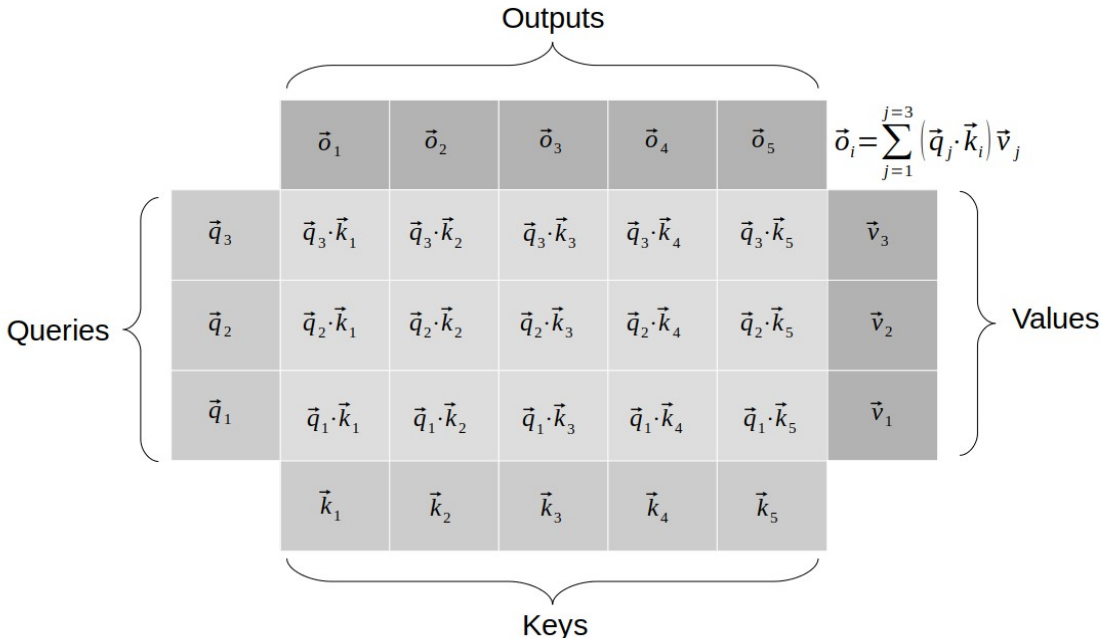

Figure 7: Standard attention layer

Provided two sets of inputs (which can be of variable size), $\vec{u}_k$ and $\vec{w}_q$, the attention layer computes 3 sets of vectors – keys, $\vec{k}_i$, queries, $\vec{q}_i$, and values, $\vec{v}_i$ – using 3 separate sets of weights – W_k, W_q, W_v – arranged in a MLP structure.
For example, the keys and queries can be expressed as $A(W_k \vec{u}_{k_i} + \vec{b}_k)$ and $A(W_q \vec{w}_{q_i} + \vec{b}_q)$, where A denotes a non-linear activation function, while the values can be obtained in a similar manner from a combination of keys and queries.
After these vectors have been computed, the inner product between each key and query is calculated and **SoftMax** activated for each key separately (along the column direction). Next, the output vectors, $\vec{o}_i$, are evaluated by taking a linear combination of the values with the normalised key coefficients, $\vec{q}_j \cdot \vec{k}_i$.
What makes this type of layer attractive is the fact that the model can learn which input features are more significant in the determination of the target output, from which the denomination "attention" layer or mechanism.

In the development of the SOAP predictor model and in the modification of the node features encoding process, the attention mechanism shown in **fig. 8** was used.
Keys and queries were obtained using a MLP acting on the concatenation of the node features and some global system-wide vector. While each key had its own set of values obtained through the application of a MLP acting on the concatenation of the given key with the queries.
The remainder of the layer operates like the one described in **fig. 7**.

Attention layers are very useful for their flexibility at picking up important input features, but, due to the large number of non-linearities, they can be unstable if not properly conditioned. Conditioning might involve batch normalisation, the use of skip connections between keys and outputs or reinstating the inputs at each attention layer if many of them are stacked one after the other.

In the following sections, whenever reference is made to attention, the attention mechanism in **fig. 8** is implied.

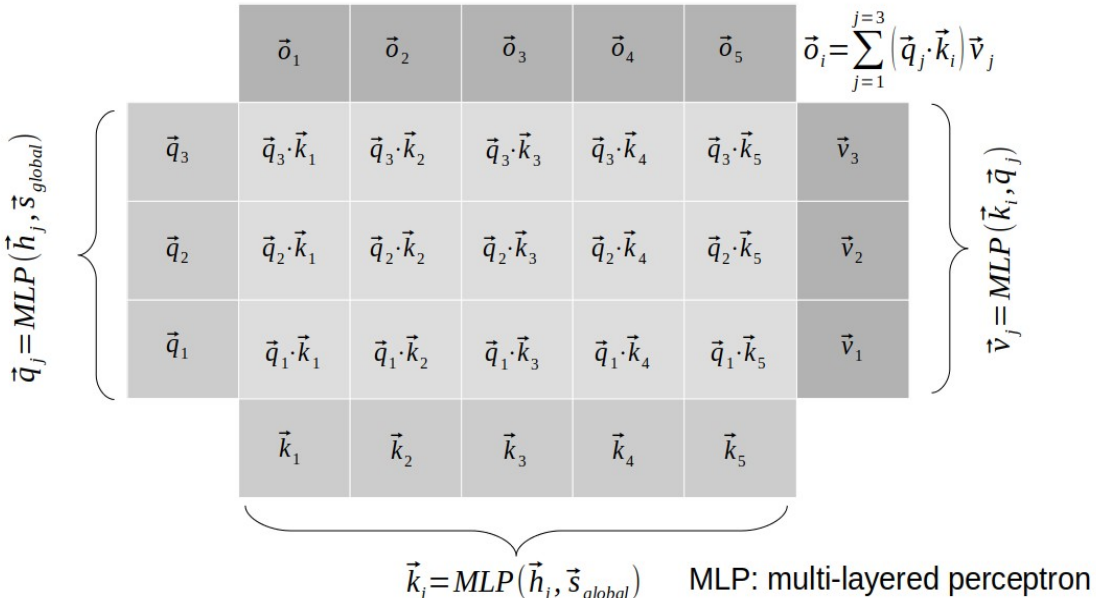

Figure 8: Attention layer used for the SOAP predictor and for the node features processing in the modified ASD model.

4.1.1 The SOAP predictor model

The SOAP predictor model that is presented thereafter is intended to be a prototype to assess the feasibility of including such a block into the ASD model. At this stage it is effectively a stand-alone block whose training procedure carries little resemblance to the one designed for the original ASD model. Furthermore, the SOAP object instantiation used here is different from the one that is found in the modified ASD model. In what follows, it is assumed that: $l_{max} = 3$, $n_{max} = 4$, $r_{cut} = 4$, $\sigma/\alpha = 1$, so that the SOAP vector for a chemical environment consists of 64 entries, 16 entries for each l-component.

The SOAP predictor block is made up of 3 different components:

- a net that computes a system-wide encoding involving the locations of the centre atoms, the locations of the other nodes, the *centre atoms-other nodes* distances and the *origin-centre atoms-other nodes* angles;
- A net that computes a system-wide encoding involving the SOAP vectors for all identified chemical environments;
- a set of SOAP predictor heads that estimate the various l-components of the SOAP vector of each chemical environment for each centre atom.

These different parts of the model are shown in **fig. 9**, **10** and **11**. This model design assumes an iterative procedure for building up the graph so that two types of SOAP predictor heads are available: one for chemical environments that had already been classified, and one for new chemical environments.

Fig. 9 displays the structure of the system-wide encoders. Assuming that the current graph has *N* nodes and *E* chemical environments, the top branch takes an input of size (*N, N, 8*), that is for each centre node there are *N* vector inputs, one for each possible *centre atom-node* combination (including the centre atom itself), of dimension 8, where the entries are: the centre atom coordinate, the node coordinate, the distance *centre atom-node* and the angle *origin-centre atom-node*.
This input is passed through 3 layers of self-attention – that is an attention mechanism in which the key and query inputs are the same – which mix the input vectors and at the same time expand their

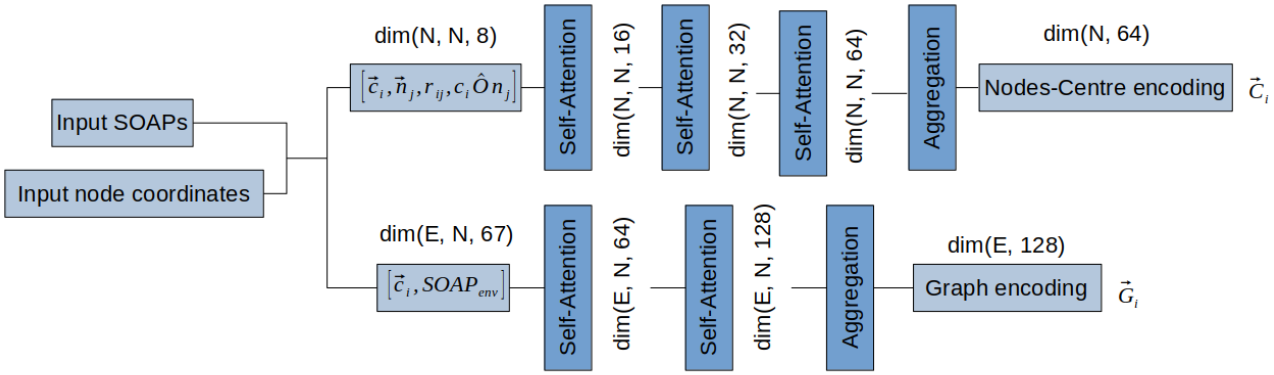

Figure 9: Global encoders blocks of the SOAP predictor model: (top branch) centre-nodes encoder, (bottom branch) chemical environment encoder.

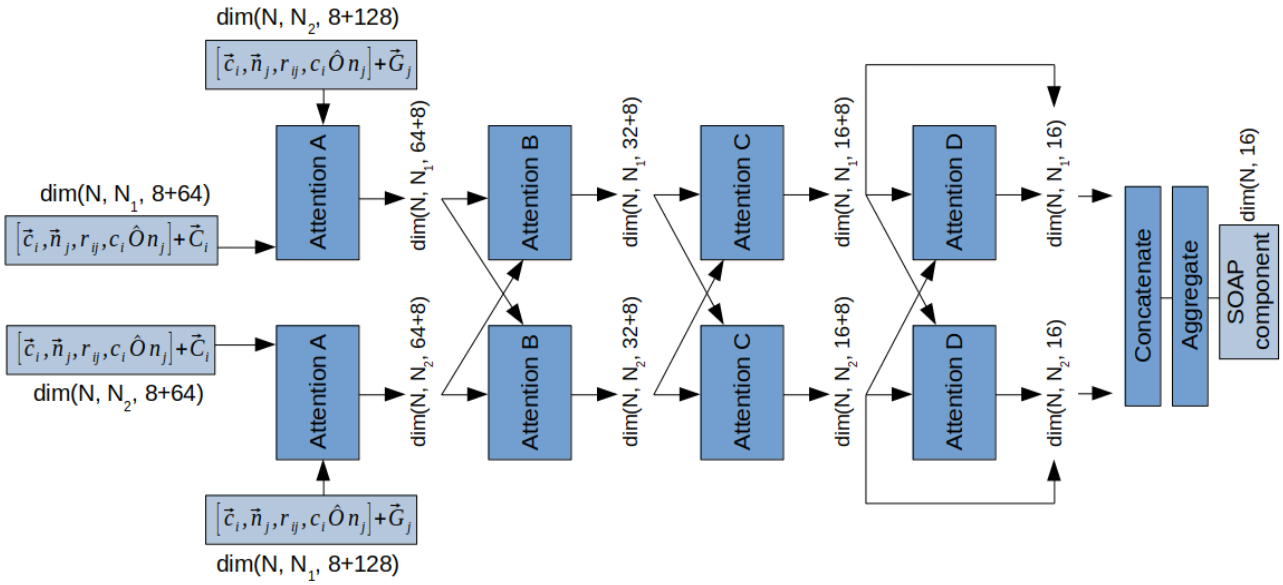

Figure 10: SOAP predictor head for previously classified chemical environments.

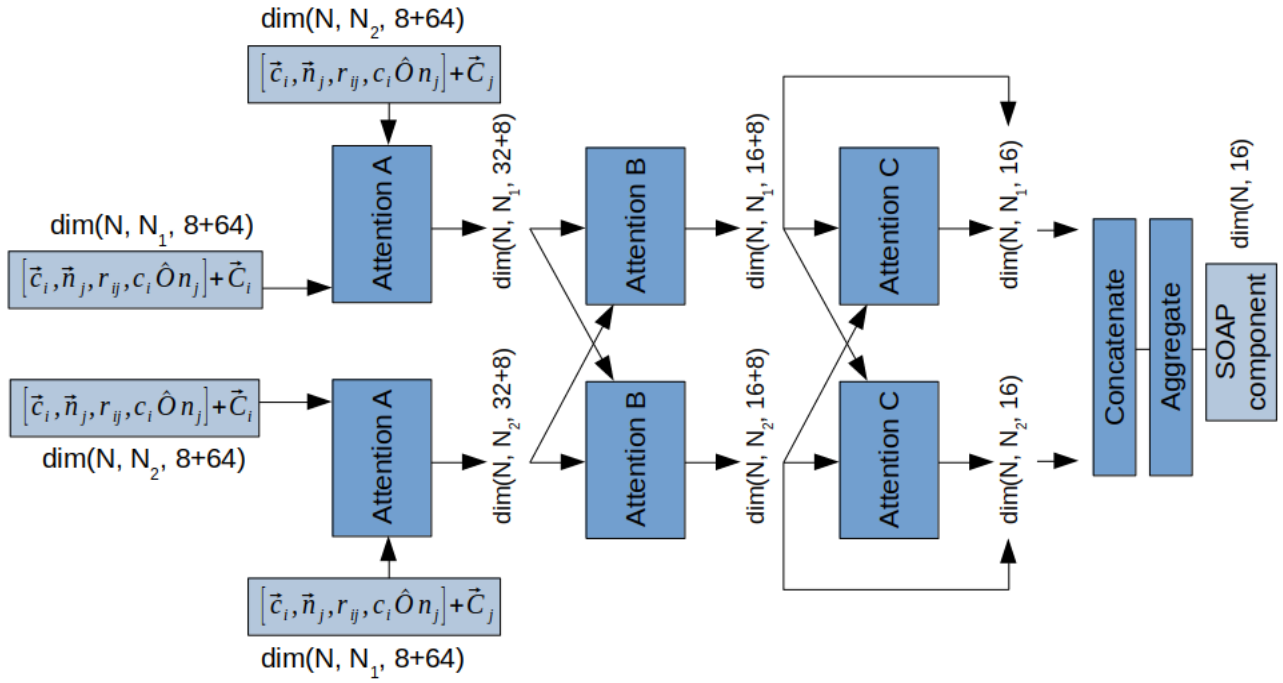

Figure 11: SOAP predictor head for newly classified chemical environments.

size from 8 to 64. After the last layer, the expanded vectors are aggregated using a **gated aggregation** process:

$\{\vec{u}\}_{i=1}^{N}$ input vectors

$$\vec{g}_i = \sigma(W_g \vec{u}_i) \qquad \text{gate vector with entries in the range (0, 1)}$$
$$\vec{v}_i = W_v \vec{u}_i \qquad \text{value vectors}$$
$$\vec{v}_{aggr} = \sum_{i=1}^{N} \vec{g}_i \odot v_i \qquad \text{final aggregated output}$$

where the dotted circle denotes an entry-wise product.
Finally, the output has dimension (*N, 64*), that is a vector encoding (thereafter referred to as *centre-nodes* encoding) per centre node.

The same type of procedure is run to generate the SOAP-based system wide encoding. The input here has dimension (*E, N, 67*), that is for each different chemical environment there are *N* input vectors composed of the node coordinate and the relevant SOAP vector for that node. These vectors go through 2 rounds of self-attention and then are combined using the gated aggregation discussed above. The results are *E* system-wide encodings each of size 128.
These encodings will be used as inputs for the SOAP predictor heads.

The architecture of the SOAP predictor heads is based on **multi-layered attention with two-channel symmetrisation**. This means that there are two input attention layers such that the key inputs of one are the query inputs of the other. The subsequent layers are constructed so that the key inputs of a layer are the outputs of the previous layer in that channel, while the query inputs are the outputs of the previous layer but in the other (symmetric) channel.
Considering the SOAP predictor head for previously classified environments, the basic input has size (*N, N_i, 8*), which is the same type of input used in the centre-node encoder, but the nodes involved belong to a single atomic class. In this context, the dimension *N* is the centre atom dimension, while the N_i dimension is the class-specific node dimension.
The key and query inputs of the attention layers are then one such tensor – where the N_i dimension may or may not correspond to the same atomic species – concatenated with the relevant *centre-nodes* encodings in the key input and with the environment-specific encoding (which will depend on what chemical species the N_i dimensions correspond to) in the query input. The two channels take these inputs in reversed order as shown in **fig. 10**.
After the input layer, there are three more layers of attention (whose workflow has already be described), the last of which presents a skip connection.
As the vectors propagate through the attention head, their size is reduced to the l-component SOAP size as shown by by the dimension labels in the diagram.
Finally, the output vectors are concatenated along the N_i dimension and gate aggregated to produce an output (*N, 16*), which is the SOAP l-component for each centre atom in the graph for the chemical environment determined by the choice of atomic species in the N_i input dimensions.

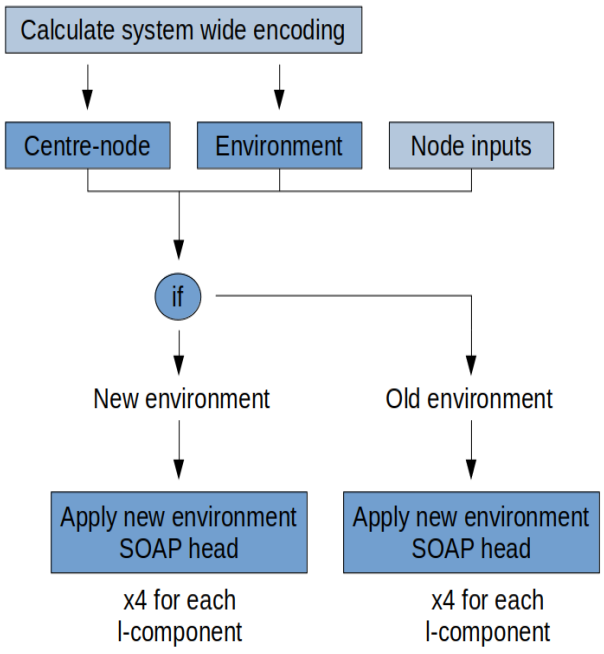

Figure 12: Workflow of the SOAP predictor model.

The SOAP predictor head for newly classified environments presents a similar architecture with the only differences being the lack of an attention layer and the use of the *node-centres* encodings as system input in both keys and queries.

The overall model workflow then proceeds according to the flow diagram displayed in **fig. 12**.

4.1.2 The training procedure

The training procedure was designed keeping in mind the mathematical properties of the SOAP descriptor and to ensure a fast execution of the model thanks to extreme vectorisation.
The flow diagram of the training procedure is shown in **fig. 13**.

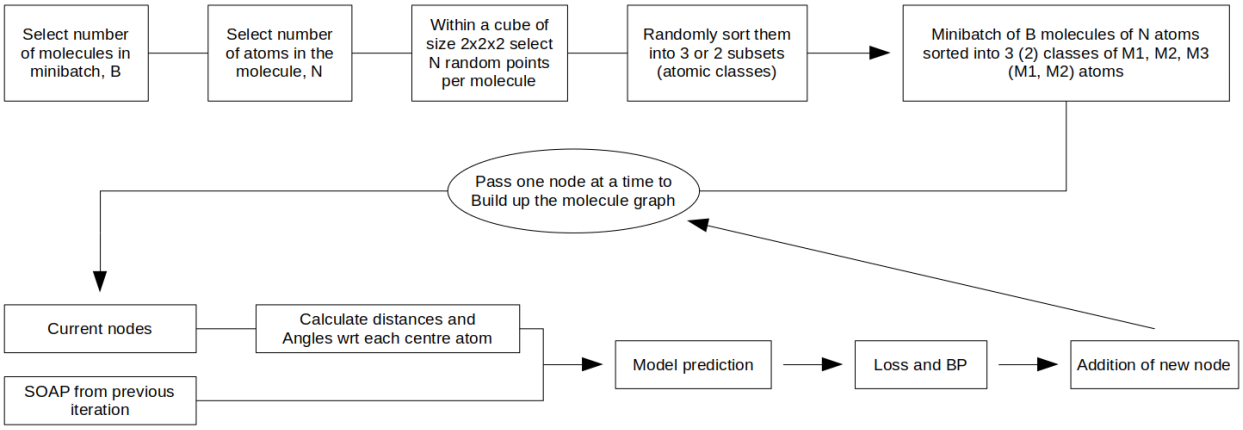

Figure 13: Flow diagram of the training procedure of the SOAP predictor model.

A training step proceeds as follows:

1. a mini-batch and molecule sizes, B and N, are selected
2. for each molecule in the batch, N random points within the cube $[-1,1]\times[-1,1]\times[-1,1]$ are sampled;
3. the N nodes in the molecular graph are divided into 2 or 3 classes such that $M_1+M_2\,(+M_3) = N$. From this partition, the label of each node in the molecular graph is decided. The node class assignment is the same for all molecules in the mini-batch;
4. the nodes are passed one at a time to the SOAP predictor – i.e. the graph for each molecule in the mini-batch is built iteratively like in the ASD model – and the SOAP descriptors for each chemical environment for each intermediate graph are estimated;
5. as the graphs are built up, the SOAP vectors for the current graphs are predicted, the MSE loss is calculated and an optimisation step is performed. After this, a new node is added to the graphs until all the graphs are completed;
6. between each node addition, teacher forcing is applied by substituting the predicted SOAP with a noisy version (the amount of noise is controlled by a variable parameter) of the SOAP label. The probability of teacher forcing can be varied depending on the training phase: it can be as high as 0.9 at the beginning of training, and drop to 0.5 during the last epochs.

This type workflow was devised to meet two requirements:

1. mimicking the workflow of the ASD model in which the predictions about the nodes are made as the graphs are built up;
2. ensuring that the input graphs all had the same number of nodes in order to simplify the batching of the inputs.

The number of atomic classes was restricted to at most 3 since this was the minimum number of classes needed to cover all possible scenarios of different SOAP environments, namely:

- the two node inputs as well as the central atom belong to the same class;
- the two node inputs belong to the same class, but the central atom belongs to a different class;
- the two node inputs belong to different classes and the central atom belongs to one of them;
- the two node inputs belong to different classes and the central atom does not belong to either of them.

Ultimately, the aim of this training procedure was to train the SOAP predictor heads as if they were some sort of mathematical operator. In fact, the mathematical definition of the SOAP descriptor involves the atomic classes just as labels, not as calculation entries, which implies that, so long all environmental instances are accounted for, the ability of the SOAP predictor head to estimate SOAP for real molecules should not be hindered.

In the same spirit of following the mathematical definition of the SOAP vector, the model contains 4 different SOAP predictor heads, one for each l-component. After all these components are in principle independent form each other.

4.1.3 Results and discussion

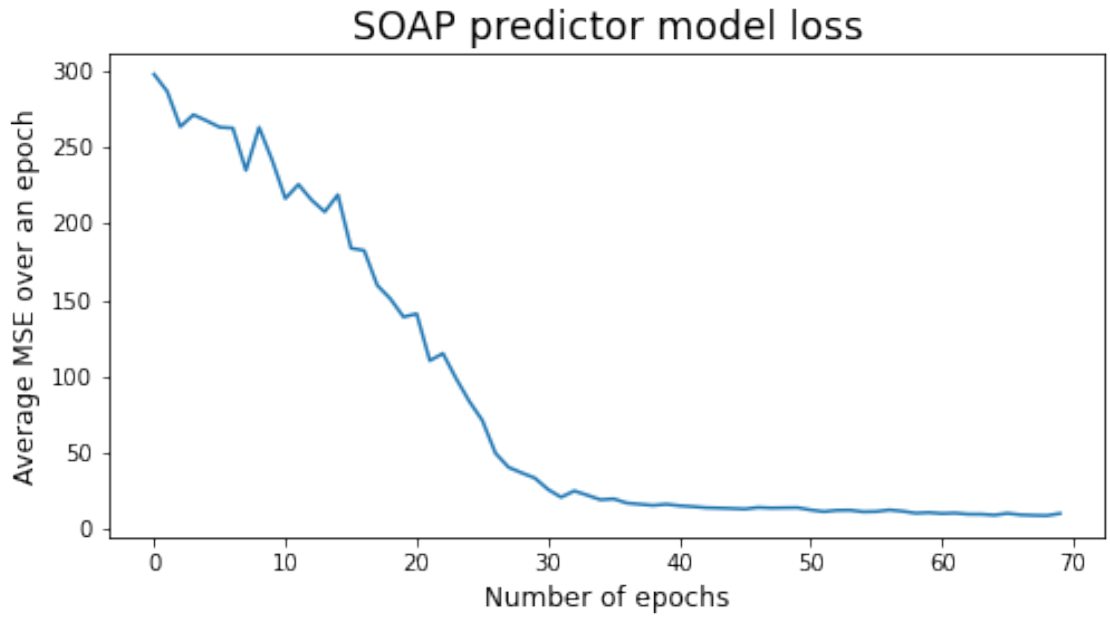

Figure 14: MSE loss averaged over the number of graph construction iterations VS epoch number for the SOAP predictor model: the MSE loss is averaged over the number of mini-batches in an epoch, here 250 mini-batches of 100 10-atom molecules.

The SOAP predictor model was trained as explained in the previous section for 70 epochs of 250 mini-batches of 100 10-atom molecules. The total execution time was around 20 hours, but it should be underlined that each mini-batch corresponds to 10 model executions as the graph building process is simulated in the model workflow.

Fig. 14 shows the MSE loss firstly averaged over the number of graph building model executions, and secondly averaged over the number of mini-batches in an epoch. The loss starts at around 300 and drops to 10.47 at the last epoch. Since new data are generated at the start of each graph building process, training loss and validation loss are the same in this situation.

The model performance at the last epoch translates into an error of approximately 3.23 per entry in the SOAP vector, which is not ideal for the utilisation of this block in the ASD model. In fact, the presence of such a large error can result in the instability of the ASD model if the output SOAP vectors are used as inputs for the next model iteration. The reason is that random unpredictable variations in the error can significantly affect the node classification and edge prediction tasks if the SOAP vectors are used as node features inputs.

The MSE loss shown in **fig. 14** corresponds to a SOAP predictor model that has already been partially optimised, for example through the use of deeper nets and the introduction of normalisation layers. Nevertheless, the basic structure had not been altered from the one outlined in **section 4.1.1**, which would imply that a tweak in the model architecture can probably improve the performance of the model.

A modification that would be beneficial to the model is to detach the addition of the first atom to the graph from the addition of the remaining atoms. This necessity arises from the fact that the SOAP vector for a single atom is fixed by the initialisation of the SOAP object (SOAP is a local descriptor so it does not depend on the atom coordinate). Hence, attempting to predict this vector using the SOAP predictor heads – which are then used for all other graph instances – would bias their weights towards predicting only this vector with high accuracy, thus worsening the performance on the other graph instances. By examining the losses for each node addition this behaviour is unsurprisingly observed: the loss on the first node addition amounts to as little as 0.001, while for the successive node additions the loss jumps to the size of unity.
The identification of other useful modifications requires further study.

Because of the low performance of this block, the SOAP vectors used as node feature inputs in the MPNN in the ASD model are calculated using the SOAP object of the DScribe library. This approach does not allow to BP though a local graph feature predicted by the model, but it is still useful to investigate whether the addition of such feature into the information processing mechanism of the model could result in an improvement of the model performance.

4.2.1 Modification to the ASD model

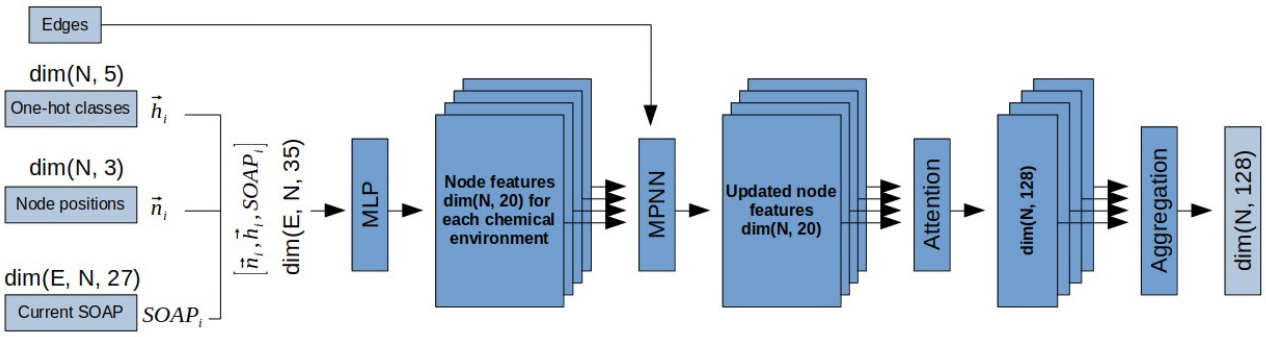

Figure 15: Node features processing block in the modified ASD model: multi-layered message passing over the different chemical environments.

Given the remarks discussed in **section 4.1.3**, the ASD model was modified as shown in **fig. 6** with the only difference being the absence of the SOAP predictor block. Therefore, the SOAP vector inputs required for the node features initialisation were calculated using the built-in method of the SOAP object in the DScribe library.

The processing of the node features is handled by the network component displayed in **fig. 15**. The input of this net consists of the position, the one-hot vector and the SOAP vectors of every possible chemical environment for each node in the (current) graph. Assuming a graph of N nodes, these inputs have dimensions (*N, 3*), (*N, 5*) and (*15, N, 27*) (see **section 2** regarding the definition of the SOAP descriptor), so that upon concatenation along the N dimension they generate a node initialisation input of size (*15, N, 35*).
This tensor is passed though a double layered MLP to produce initialised node features of size (*15, N, 20*) which are the input of the MPNN. The GNN then performs message passing separately over the features corresponding to the different chemical environments so that the output, after a certain number of rounds of message passing, still has size (*15, N, 20*).
Next the 15 features of each node undergo 2 rounds of self-attention to expand their size from 20 to 128. In this context, the rounds of self-attention are intended to select which of the 15 node features carries the most significant information for the purpose of classifying the nodes or predicting the edge connections.
After the attention layers, the 15 feature vectors of each node are gate aggregated resulting in a final output of dimension (*N, 128*). These are the final node hidden state vectors which can be further

aggregated to produce a graph encoding vector for the node classifier or they can be used as inputs for the edge predictor.

Since these two tasks are very different in nature, **the parameters of the initialisation MLP, the attention layers and the gate aggregator are not shared** in order to give sufficient freedom to the system to select the best parameters for the two operations. In the same spirit, the number of MPNN rounds is different in the two instances of the model: 3 rounds for the node classification task and 6 rounds for the edge prediction task. Also in this situation, **the parameters of the MPNN are not shared**.

The break-up of the original ASD model into effectively independent blocks, which do not share parameters, results in almost a doubling of the number of model parameters (from 600,000 to about one 1,100,000), but this might lead to a performance improvement.

Giving a final general glance at the modified ASD model, the workflow proceeds now as follows:

1. the SOAP vectors and the graph information from the previous iteration are used to produce a graph encoding vector by passing those inputs through the node features processing block discussed in this section;
2. the graph encoding vector along with the CNN encoder layers outputs are used to generate an AFM image encoding;
3. the image encoding along with the new node position are fed to the node classifier which predicts the class of the new node;
4. this information is used to update the SOAP vectors of the previously classified nodes and to compute the SOAP vectors of the new node;
5. both the old nodes and the new node features are initialised again: the old nodes features are processed as in point 2, while the new node features are left untouched;
6. the updated node features and the new node features are used by the edge predictor block to identify the edge connections to the new node;
7. the outputs of the net are fed back to the model to start the next iteration.

4.2.2 The training procedure

The training procedure implemented to train the new model is identical to the one described in [1]. The dataset and the type of loss function are the same as well. A short summary of the key points is provided here, but the interested reader is invited to consult the original publication.

The database used for training the model consists of a train, a validation and a test dataset which contains 180,000, 20,000, 35,553 samples respectively.
Each sample is composed of a simulated AFM image, an xyz file (the reference for the atomic positions) and a reference graph.
The inputs for the CNN part of the model are created by applying white noise, random rotation and translation and random crop to the AFM images. The inputs of the GNN branch are produced by firstly applying the same transformations to the reference graphs and secondly by removing a random number of nodes and edges.
The model is then trained to predict the positions, the class and the edge connections of the missing atoms.

Due to the randomness of the transformations and the fact that a single graph requires a sequence of predictions, the dataset is augmented beyond the absolute number of samples which is useful to prevent over-fitting especially when the number of free parameters is very large.

After training the model on the train and validation datasets, its accuracy is evaluated on the test dataset. Since the molecules in this set do not appear in the training sets, the statistical information that is extracted from them is a good representation of the actual performance of the model.
Various statistics are then calculated from the outputs of the model on the test samples. The most informative amongst them are the confusion matrices (precision and recall diagrams) for the node classification and the edge prediction tasks. Other statistical indicators include the precision and the recall for the aforementioned tasks binned according to the molecule size. However, these plots are less interesting because they mostly confirm the intuitive expectation that the model performance would drop as the graph size increases due to the presence of fewer data points for the model to learn from.

4.2.3 Results and discussion

Fig. 16 shows the traces of the training and the validation losses over 50 epochs of training.

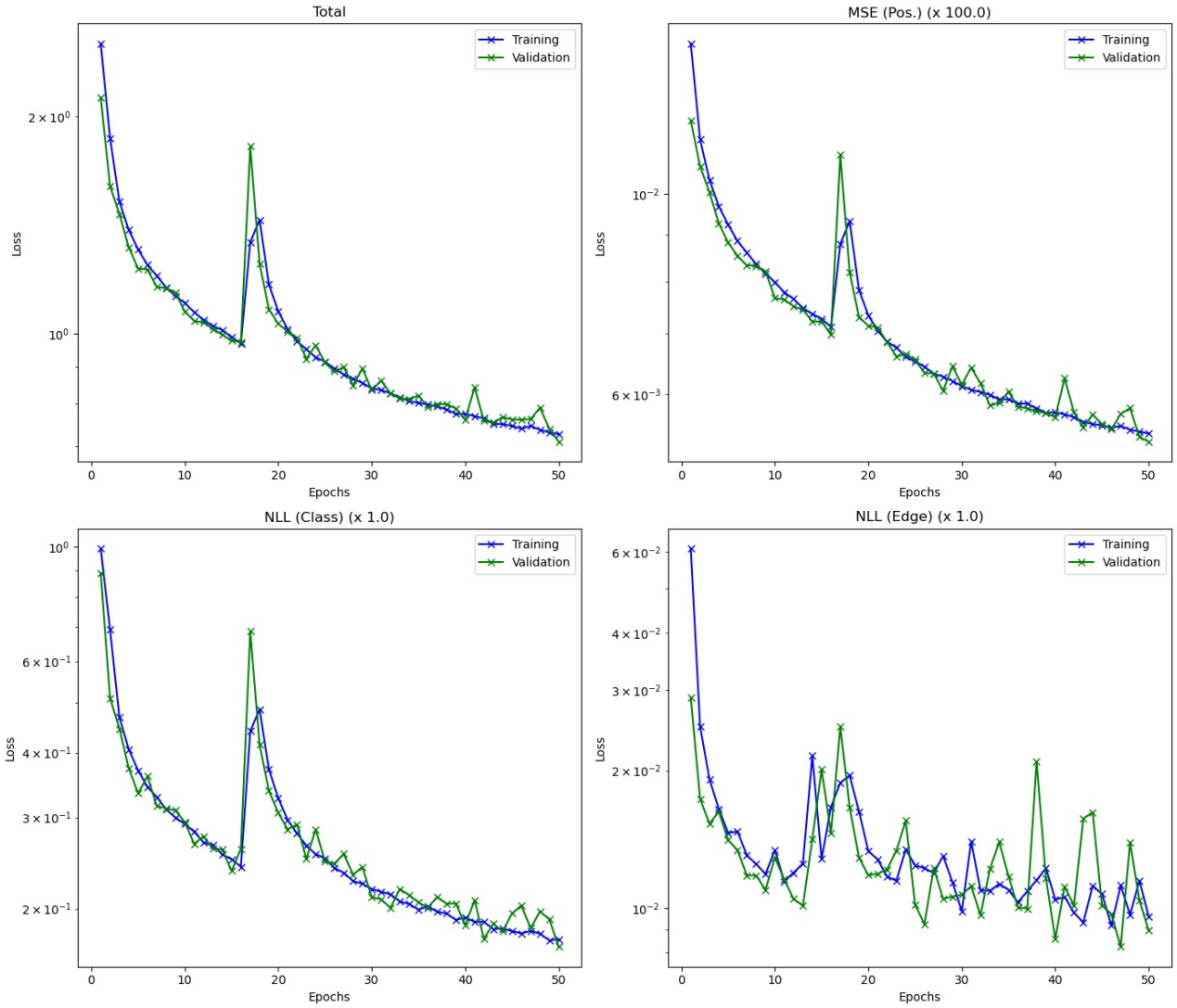

Figure 16: Loss diagrams for the model with no shared parameters between the node classification and edge prediction tasks (Message Passing is turned on). Starting from the top right and moving in the clockwise direction: MSE loss for the atom position (CNN part of the model), cross-entropy loss for the edge prediction task, cross-entropy loss for the node prediction task and the total loss (where the MSE is weighted by a factor of 100 [1]).

The MSE loss of the CNN and the cross-entropy loss of the node classifier vary smoothly except for a large peak around epoch 20, which is damped down during the subsequent epochs. On the other hand, the binary cross-entropy loss of the edge predictor, despite having a slow decreasing trend, fluctuates a lot, hence it does not decrease as much as in the version of the ASD model discussed in **appendix B**.

This behaviour is reflected in the confusion matrices for the node classification and the edge prediction tasks.

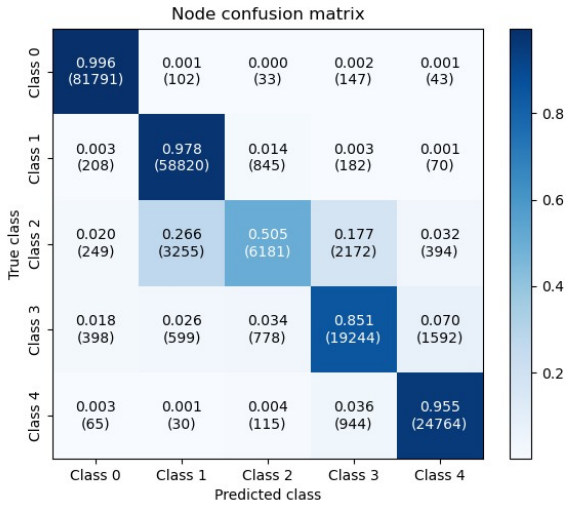

Figure 17: Confusion matrix for the node classification problem, epoch 50 (random order graph construction).

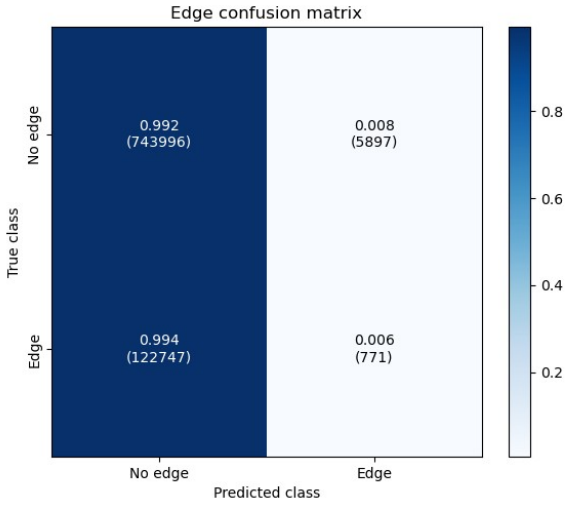

Figure 18: Confusion matrix for the edge prediction problem, epoch 50 (random order graph construction).

The node classifier performs slightly better than the original model does at the classification of class 0, 1, 3 and 4 atoms. Yet, its precision drops significantly, from 0.563 [1] to 0.505, in the context of class 2 atoms.

By contrast, the model precision has importantly decreased in the edge prediction task. The confusion matrix of **fig. 18** clearly indicates that the model at its current state of training does not predict any edges.

Curiously the model was performing much better at this task when message passing was deactivated (**appendix B**). Therefore, having message passing working for the node classification and turning it off for the edge prediction might improve this aspect of the model.

The noisy evolution of the model parameters in this version of the ASD model probably lies in the fact that, in this form, the system has to learn not only the message passing parameters for the atomic coordinates and classes, but also for the SOAP descriptor, which would imply that the number of required training epochs might be greater that 50 (see **appendix E**). After all, the SOAP descriptor embeds a large amount of geometrical information about the local environments of the atoms in the molecule and it is also sensitive to variations of the atomic positions since it is differentiable with respect to them.

Furthermore, it is possible that the presence of message passing followed by attention layers could on its own cause the instabilities that were observed. So, a change in the model architecture involving solely these components could likely mitigate this issue.

In addition to these points, the introduction of regularisation techniques, such as least-square regularisation, might damp down the oscillations observed in the cross-entropy loss of the edge prediction task since it would bias the system towards smaller weights which should result in a smoother behaviour of the model. This can be especially useful here given the differentiable nature of the SOAP descriptor.

Another way to train the model to correctly predict the edges could be turning off the gradients for all other tasks (the CNN and the node classifier), while fine tuning the edge predictor parameters only.

5. Conclusion

During the course of this project, the possibility of integrating the engineered molecular descriptor SOAP into the information processing mechanism of the ASD model was investigated.

Initially, an attempt was made to incorporate a SOAP predictor block into the system in order to facilitate the model in learning translational and rotational equivariance. Although the results for this sub-block looked promising, the still too low accuracy and the lack of performance information regarding large molecules meant that the addition of this new block to the ASD model had to wait.

Following, the research effort focused on adding the SOAP descriptor as an "a priori" calculated feature into the system. To this purpose, the development of a SOAP predictor block for the ASD model was rather useful since many questions concerning the integration of the SOAP descriptor into the molecular graph construction loop had already been addressed there.
After the ASD model was altered to process the SOAP descriptors of the atoms in a molecule as well as the original set of inputs, various numerical experiments were run. These include: the model with shared parameters for the node classifier and the edge predictor (**appendix A**), the model without message passing (**appendix B** and **C**) and the model with message passing (**section 4.2.3** and **appendix E**).
The confusion matrices for the latter two versions of the model did not show a particular improvement with respect to the original ASD model which would suggest that for this specific application giving the freedom to the system to learn what the relevant molecular features are, might be more useful.

The results are however limited in scope due to the long training time of the model, around 3 days on 4 ampere class GPUs for 50 epochs. Therefore, optimising the model, for example by separating the CNN from the graph reconstruction net, could be very beneficial for the purpose of running more numerical experiments.

References

[1] Oinonen, N., Kurki, L., Ilin, A. *et al.* Molecule graph reconstruction from atomic force microscope images with machine learning. *MRS Bulletin* (2022). https://doi.org/10.1557/s43577-022-00324-3.
[2] Electrostatic Discovery Atomic Force Microscopy, *ACS Nano* 2022, 16, 1, 89–97, November 22, 2021, https://doi.org/10.1021/acsnano.1c06840.
[3] Automated structure discovery in atomic force microscopy, Alldritt et al., Sci. Adv. 2020; 6 : eaay6913, 26 February 2020.
[4] Attention is all you need, Vaswani et al., https://arxiv.org/abs/1706.03762, https://doi.org/10.48550/arXiv.1706.03762.
[5] Beilstein J. Nanotechnol, 2021, 12, 878–901, https://doi.org/10.3762/bjnano.12.66.
[6] Oliver M Gordon and Philip J Moriarty 2020 Mach. Learn.: Sci. Technol. 1 023001.
[7] Neural Message Passing for Quantum Chemistry, Gilmer et al., https://arxiv.org/abs/1704.01212, https://doi.org/10.48550/arXiv.1704.01212.
[8] Learning Deep Generative Models of Graphs, Yujia Li et al., https://arxiv.org/abs/1803.03324, https://doi.org/10.48550/arXiv.1803.03324.
[9] Relational inductive biases, deep learning, and graph networks, Battaglia et al., https://arxiv.org/abs/1806.01261, https://doi.org/10.48550/arXiv.1806.01261.
[10] DScribe: Library of descriptors for machine learning in materials science, Lauri Himanen et al., https://doi.org/10.1016/j.cpc.2019.106949.
[11] Optimizing many-body atomic descriptors for enhanced computational performance of machine learning based inter-atomic potentials, Miguel A. Caro, 2019, https://doi.org/10.1103/PhysRevB.100.024112.
[12] Comparing molecules and solids across structural and alchemical space, Michele Ceriotti et al., Phys. Chem. Chem. Phys., 2016, 18, 13754, 10.1039/c6cp00415f.
[13] Machine Learning Prediction of Superconducting Critical Temperature through the Structural Descriptor, J. Phys. Chem. C 2022, 126, 8922−8927.
[14] Machine Learning, Zhi-Hua Zhou, Springer, https://doi.org/10.1007/978-981-15-1967-3.
[15] Atomic Force Microscopy, Peter Eaton, Paul West, Oxford University Press, 2010.
[16] Mechanism of high-resolution STM/AFM imaging with functionalized tips, Prokop Hapala, Pavel Jelínek et al., 2014, American Physical Society, http://dx.doi.org/10.1103/PhysRevB.90.085421.
[17] A direct method to calculate tip–sample forces from frequency shifts in frequency-modulation atomic force microscopy, F. J. Giessibl, Appl. Phys. Lett. 78, 123 (2001), https://doi.org/10.1063/1.1335546.

Appendix

IMPORTANT NOTE: the results shown in appendix A, B and C were obtained for a model where the MPNN was turned off. That is the the node features were just initialised and message passing was not performed.

A. Modified ASD model results prior to the separation of the shared parameters

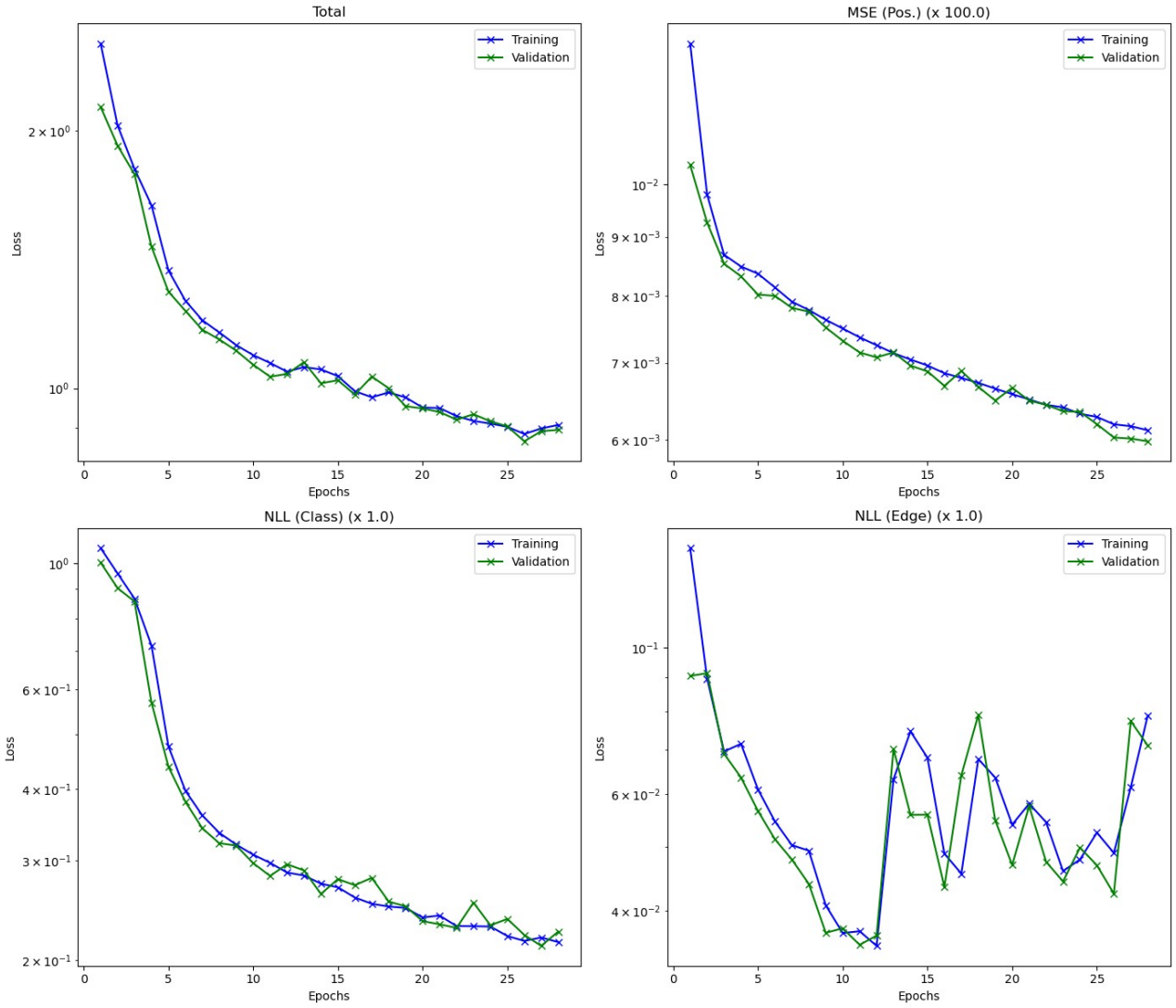

Figure 19: Loss diagrams for the model with shared parameters for the node classification and edge prediction tasks (Message Passing is turned off). Starting from the top right and moving in the clockwise direction: MSE loss for the atom position (CNN part of the model), cross-entropy loss for the edge prediction task, cross-entropy loss for the node prediction task and the total loss (where the MSE is weighted by a factor of 100 [1]).

Before the ASD model was formulated as described in **section 4.2.1**, the parameters of the node initialisation layer, the attention layers, the gate aggregator and the MPNN were shared between the node classification and the edge prediction tasks.
These operations are very different in nature and, given the rich geometrical information embedded in the SOAP descriptor, different SOAP components or environments could be more significant for a task than the other. This can result in a conflict between the performance of the node classification and the edge prediction during the optimisation phase since a set of model parameters might be good for one operation, but not for the other.
This was in fact observed while training the shared parameters version of the ASD model.
As shown in **fig. 19**, the node classification loss and the edge prediction loss (bottom two diagrams in the illustration) decrease together up until epoch 12. Beyond this point, the node classification loss keeps going down, while the edge prediction loss is affected by phenomenon that resembles ringing, that is it oscillates about a constant value.

As previously mentioned, the reason for this type of behaviour is probably the clash between different parameter requirements for the two tasks, so that it is expected that the model efficiency would benefit from having two sets of independent parameters for the two tasks (if parameters are shared, gradient descent will prioritise the task with the larger loss).

In fact, the model at the stage displayed in **fig. 19** is as efficient as (or maybe slightly worse than) the original formulation. Therefore, a larger number of parameters alongside the additional information coming from the SOAP descriptor could be essential to boost the accuracy of the model.

B. Modified ASD model results after the separation of the shared parameters

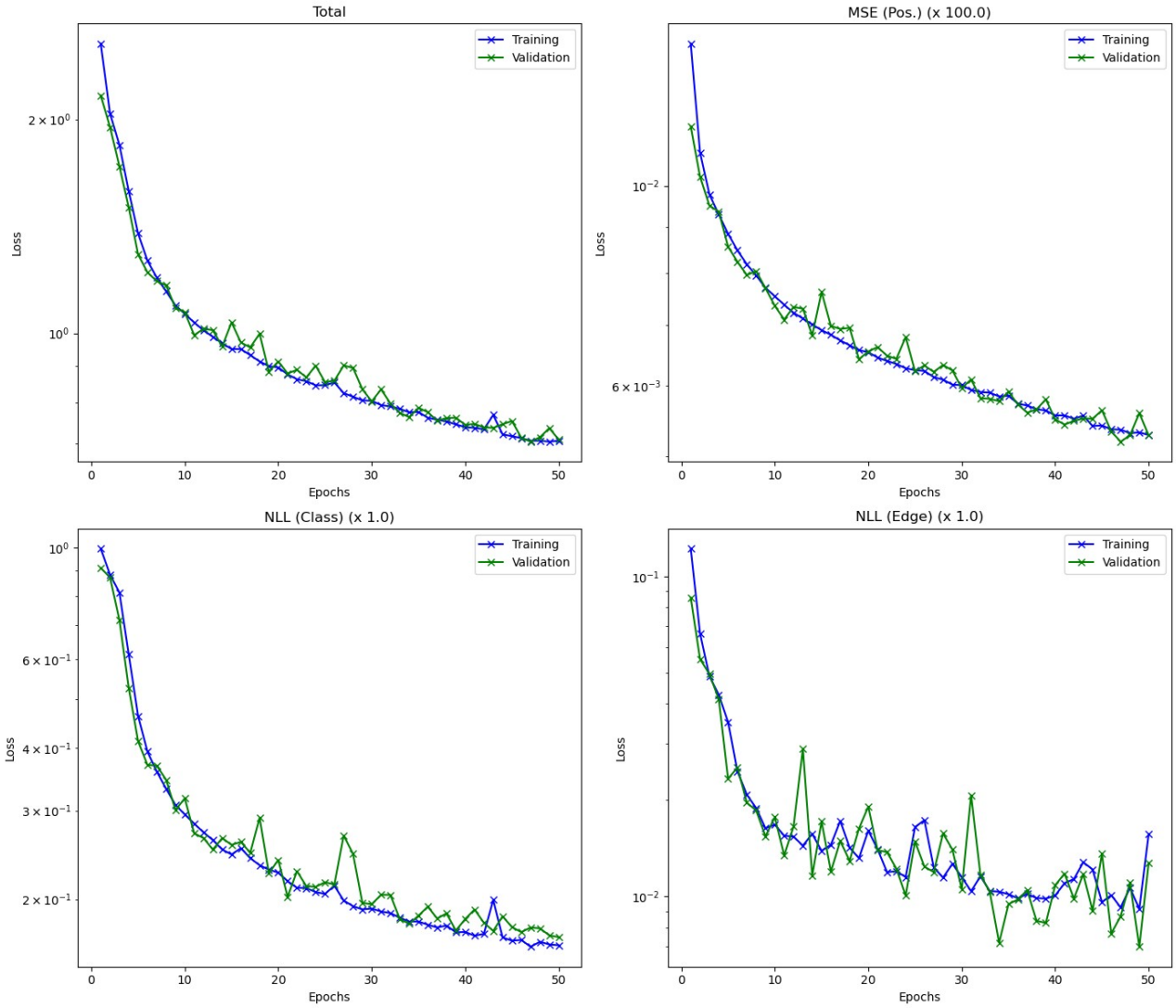

*Figure 20: Loss diagram for the model where the node classification and the edge prediction tasks do not have parameters in common (Message Passing is turned off). Refer to **fig. 19** for details concerning the individual plots.*

After the ASD model was refactored so that the node classification and the edge prediction tasks did not have any layers with parameters in common, training was performed a second time.
The results shown in **fig. 20** demonstrate the validity of the observation made in **appendix A**. Separating the two tasks completely indeed improves the model learning capacity: ringing is removed and the edge prediction loss showcases a decreasing trend almost throughout training.
Nevertheless, it should be underlined that the edge prediction loss still fluctuates significantly and its trace contains wide regions where it increases for 2 or more consecutive epochs before it curves down again.

This unwanted behaviour probably arises from two factors:

1. the stability issues related to attention layers;
2. the fact that message passing is turned off: since information is not transferred across nodes, the model does not directly process information about the nodes surrounding a given node. The only information in this sense comes form the SOAP descriptor which by construction characterises the geometry surrounding a given atom.

Point 2 is particularly important because, one one side, it implies that message passing is essential to the characterisation of molecular graphs and its implementation would ensure a smoother variation of the model weights thus damping the fluctuations in the loss, while, on the other side, it proves that the SOAP descriptor on its own possesses enough geometrical and chemical (about the node classes) information to substitute message passing, at least in part.

Expanding on this last statement, **fig. 20** shows that the total validation loss at epoch 47 (~0.7039) is lower than the one registered at the end of the training of the original model (~0.78). The same point is substantiated also by the confusion matrices, **fig. 21** and **22**, that were obtained when the model (weights of epoch 49, validation loss ~0.735) was applied to predict the molecular graphs of the molecules in the test set.

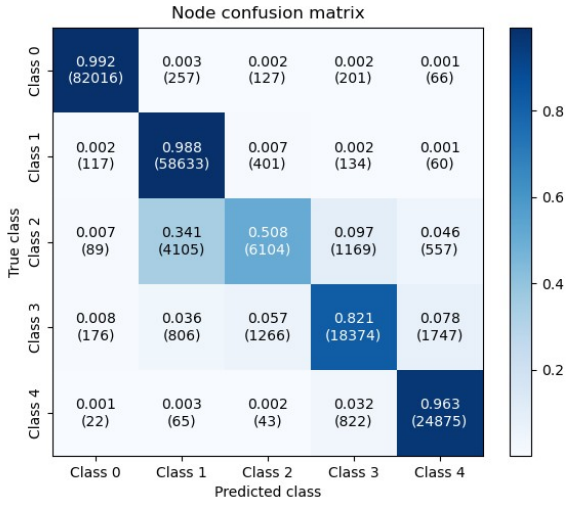

Figure 21: Confusion matrix for the node classification problem, epoch 49 (random order graph construction).

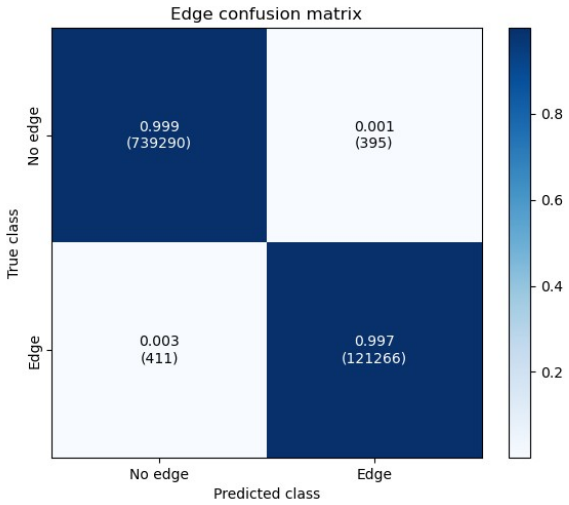

Figure 22: Confusion matrix for the edge prediction problem, epoch 49 (random order graph construction).

Comparing the diagonal values (the precision) to the ones in **section 2 of reference [1]**, there is a slight improvement of those cases in which the original model was already performing well, yet there is a slight worsening of the class 3 classification precision, and a more pronounced drop in the class 2 classification precision.

This can either be a hallmark of over-fitting (after all the number of parameters has been doubled), or a problem related to the absence of message passing between the nodes. Nonetheless, given that the performance has not degraded considerably despite the MPNN being off, the outcome of model testing indicates that the SOAP descriptor can potentially be very useful in boosting the efficiency of the model once message passing is restored.

C. Confusion matrices for the model at epoch 47

The performance of the model was also tested for the parameter settings of epoch 47, the epoch with the lowest validation loss (~0.7039). The outcome of testing is reported in the confusion matrices in **fig. 23** and **24**.

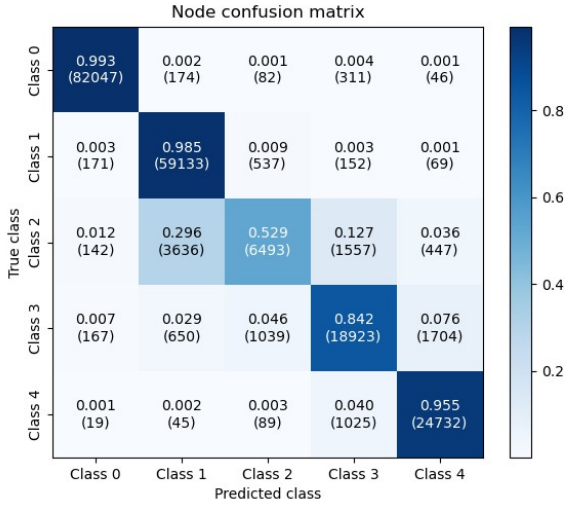

Figure 23: Confusion matrix for the node classification problem, epoch 47 (random order graph construction).

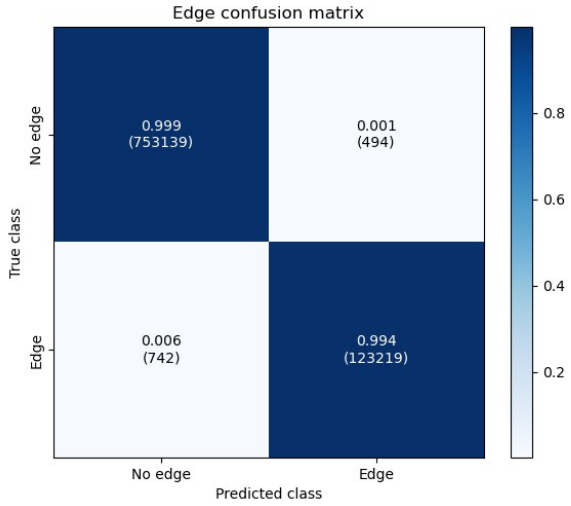

Figure 24:Confusion matrix for the edge prediction problem, epoch 49 (random order graph construction).

In comparison to the same confusion matrices for the model at epoch 49, the precision in the well performed tasks has slightly decreased, while the precision in the class 2 and class 3 classification sees a considerable improvement. The model at this stage performs worse than the original one only in the classification of class 2 atoms, while the edge prediction accuracy is comparable.

The jump in the precision of the badly performed tasks coupled with a slight worsening in the well performed tasks is a result of the fact that gradient descent favours tasks with higher loss since the gradient associated to them is usually greater.

D. Modified training procedure for the SOAP predictor

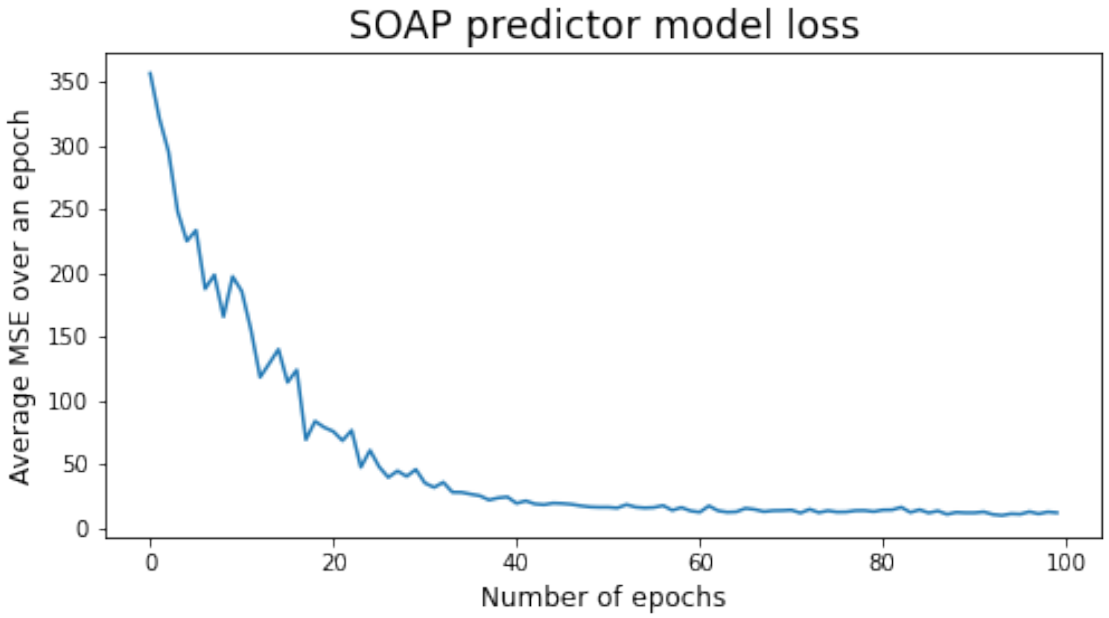

Figure 25: MSE loss averaged over the number of graph construction iterations (9 in this case) VS epoch number for the SOAP predictor model: the MSE loss is averaged over the number of mini-batches in an epoch, here 100 mini-batches of 600 10-atom molecules.

In **section 4.1.3** the issue regarding the biasing of the weights towards an accurate prediction of the SOAP descriptor for the first atom addition was raised. Therefore, the model was trained a second time according to procedure where the chain of model predictions for a given molecule started from the second atom addition.

The model was trained over 100 epochs of 100 mini-batches of 600 10-atoms molecules each. The details of the training procedure are identical to the ones discussed in **section 4.1.2**. **Fig. 25** shows the outcome of the training.
The MSE loss at the last epoch amounts to 11.801. This result is consistent with the one obtained from the initial training procedure. In fact, in that case the first atom addition contributed almost 0 to the loss, yet the average was computed over all atom additions (i.e. 10). Consequently, the mean loss was underestimated, and when corrected for this discrepancy (multiply by 10/9) it compares much more closely to the results encountered here (it becomes ~11.64).
This means that changing the training procedure only does not affect the model performance substantially, reason why a modification of the model architecture seems necessary to improve the accuracy of this block.

E. Extra training for the ASD model

As discussed in **section 4.2.3**, the new ASD model probably requires a longer training procedure since the number of weights is greater and due to the erratic behaviour of the edge prediction loss. To prove this point, the model underwent 10 extra training epochs, so that the number of epochs totalled to 60.
The training and validation losses are displayed in **fig. 26**.

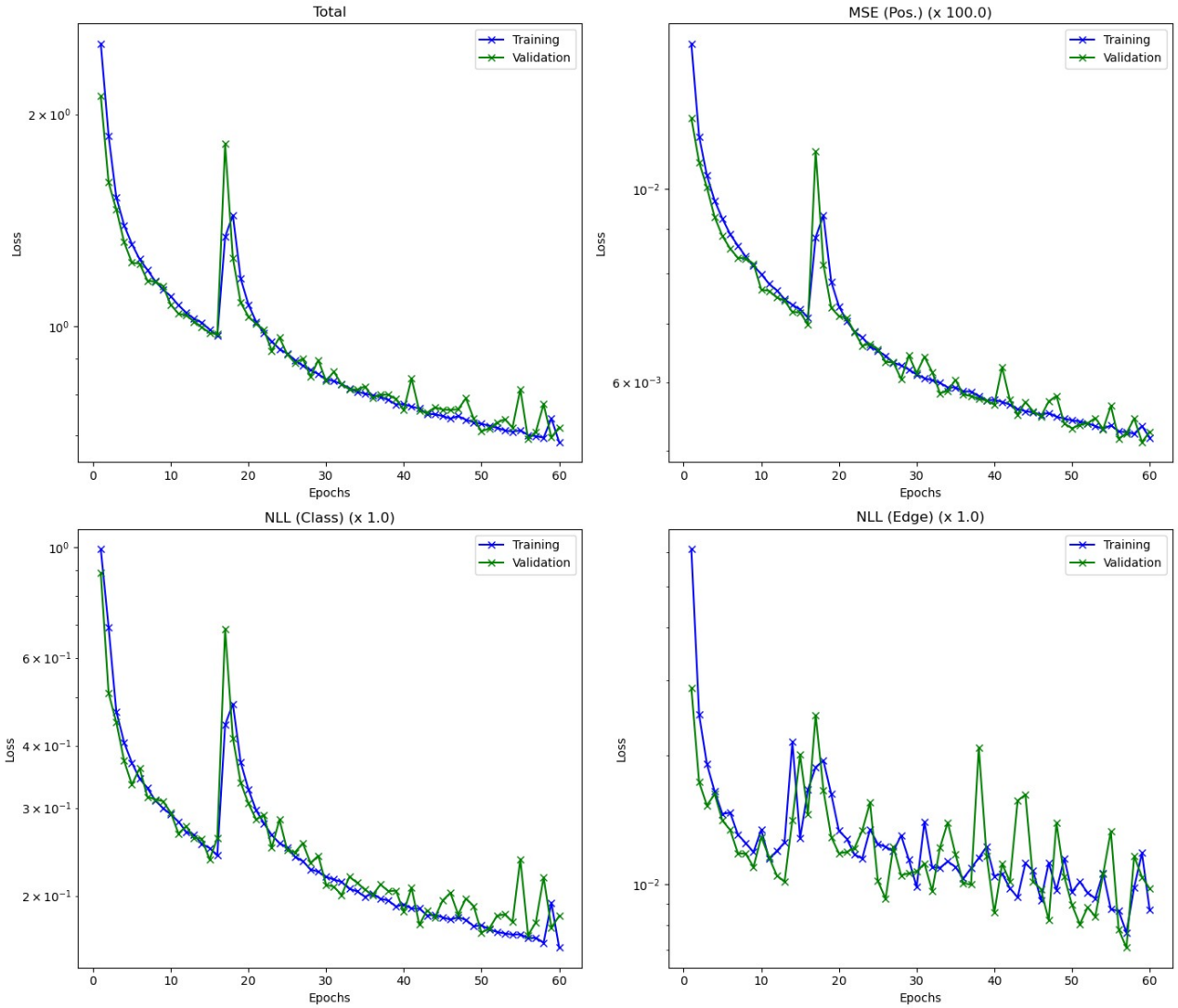

*Figure 26: Loss diagram for the model where the node classification and the edge prediction tasks do not have parameters in common (Message Passing is turned on). Refer to **fig. 19** for details concerning the individual plots*

The diagrams show that the model still has some learning capacity, but the losses for the various components of the model, even at this stage of training, are still afflicted by the instabilities

described in **section 4.2.3**. In particular, at epoch 59 both the node classification and the CNN losses have a sudden jump. The same behaviour, yet more pronounced, is seen in the edge prediction loss.

Despite the presence of these fluctuations in the loss, the additional training has reduced the validation error, which at epoch 57 amounts to 0.708095. This is reflected by an overall improvement of the accuracy of the model as shown by the confusion matrices in **fig. 27** and **28**.

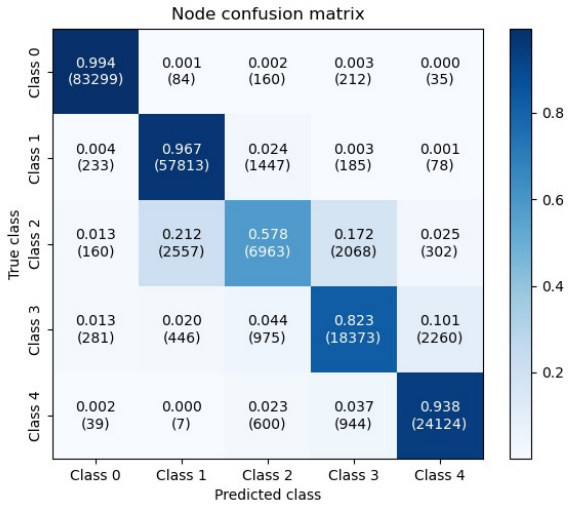

Figure 27:Confusion matrix for the node classification problem, epoch 57 (random order graph construction).

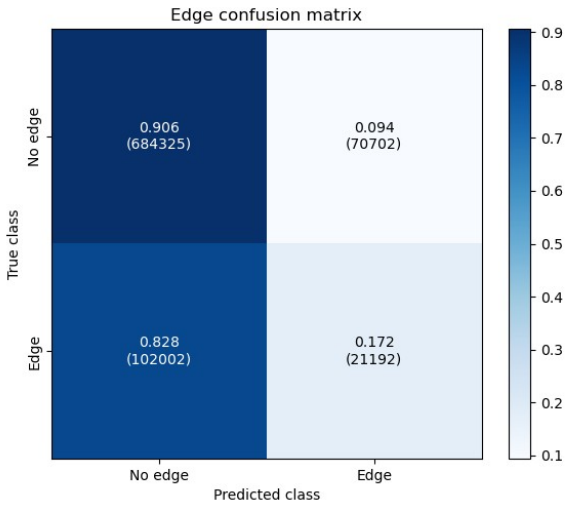

Figure 28:Confusion matrix for the edge prediction problem, epoch 57 (random order graph construction)

Fig. 27 shows for the first time a situation where the accuracy of the classification of class 2 atoms of the ASD model with the SOAP descriptor surpasses the one of the original model, while the performance of the other tasks remains approximately unaltered.
On the other hand, **fig. 28** indicates a small betterment of the model at predicting the edge connections, although the precision of this task is still rather poor.

In light of the observations made here and in **section 4.2.3**, it should be evident that the model, on one side, requires some tweaks in the architecture to stabilise its learning pattern (for instance the attention mechanisms could be made shallower), while, on the other, the amount of data used for training should be enhanced. After all, the relative rarity of molecules with Nitrogen and Phosphorous (class 2) atoms is in part to blame for the low precision of the model at identifying them.