# Supplementary Information: Molecule graph reconstruction from AFM images with machine learning

Niko Oinonen[1], Lauri Kurki[1], Alexander Ilin[2] and Adam S. Foster[1,3†]

[1]Department of Applied Physics, Aalto University, 00076 Aalto, Helsinki, Finland
[2]Department of Computer Science, Aalto University, 00076 Aalto, Helsinki, Finland
[3]WPI Nano Life Science Institute (WPI-NanoLSI), Kanazawa University, Kakuma-machi, Kanazawa 920-1192, Japan

[†]Corresponding author; Email: adam.foster@aalto.fi

## Model training and testing

The model is trained and tested on a database of molecules and corresponding simulated AFM images, which is divided into a training set of 180 000 samples, a validation set of 20 000 samples, and a test set of 35 554 samples. The database of molecules and the simulation parameters for the AFM images are detailed in [1].

For each sample, we do a constant-height CO-tip AFM simulation at 10 different heights separated by a distance of 0.1 Å, which function as the input to the model. For generating the corresponding target outputs, we need to take into account the inherent limitation in AFM imaging that the signal from deeper atoms decays very quickly with increasing depth. For planar molecules, such as the PTCDA, the target output can contain all of the atoms in the molecule, but for more 3D systems, there needs to be some cutoff for how deep atoms are included in the output. Here we use a very simple criterion, where any atom whose center position is more than a chosen threshold distance below the atom with the highest center position is cut out from the target output. Naturally, we want to maximize this cutoff distance in order to maximize the amount of information extracted from the AFM images. Through empirical testing we found a cutoff distance of 0.8 Å to be optimal. A cutoff any deeper than this resulted in a significant reduction of the model performance.

Predicting the position grid is a rather straightforward regression task, with a single input and output. However, the graph construction is an iterative process that presents some choices for the training. We choose to simplify the training process by assuming that the iteration is a Markov chain, that is, we assume that each step in the iteration is independent from the others. This allows us to train the model on individual steps of the iteration rather than having to run through the whole iteration process for a single gradient step. In practice this means that the hidden vector states for the nodes are reset on each iteration. Training samples can be generated by removing a random number of nodes from the graphs, along with their corresponding edge connections, and then have the model predict the class and edge connections of one of the removed nodes.

The selection of which of the removed nodes should be predicted presents us with two more choices. First is the choice between using the predicted atom coordinates or the reference atom coordinates. Since there is no gradient propagation from the node classification to the position grid prediction, there is no need to actually use the predicted coordinates during training. Rather, the known correct coordinates for the atom can be used, similar to the concept of *teacher forcing* used in training recurrent neural networks [2]. In practice we find the best result by using the reference coordinates with added Gaussian noise with amplitude (0.2 Å, 0.2 Å, 0.05 Å) in the (x, y, z)-directions.

The second choice in the node selection is the order in which the nodes are selected for addition to the graph. Perhaps the simplest choice is to use a completely random order, but a previous study in molecule graph generation found benefit in some cases from using a fixed ordering based on the SMILES string for the molecule [3]. Since in our case we

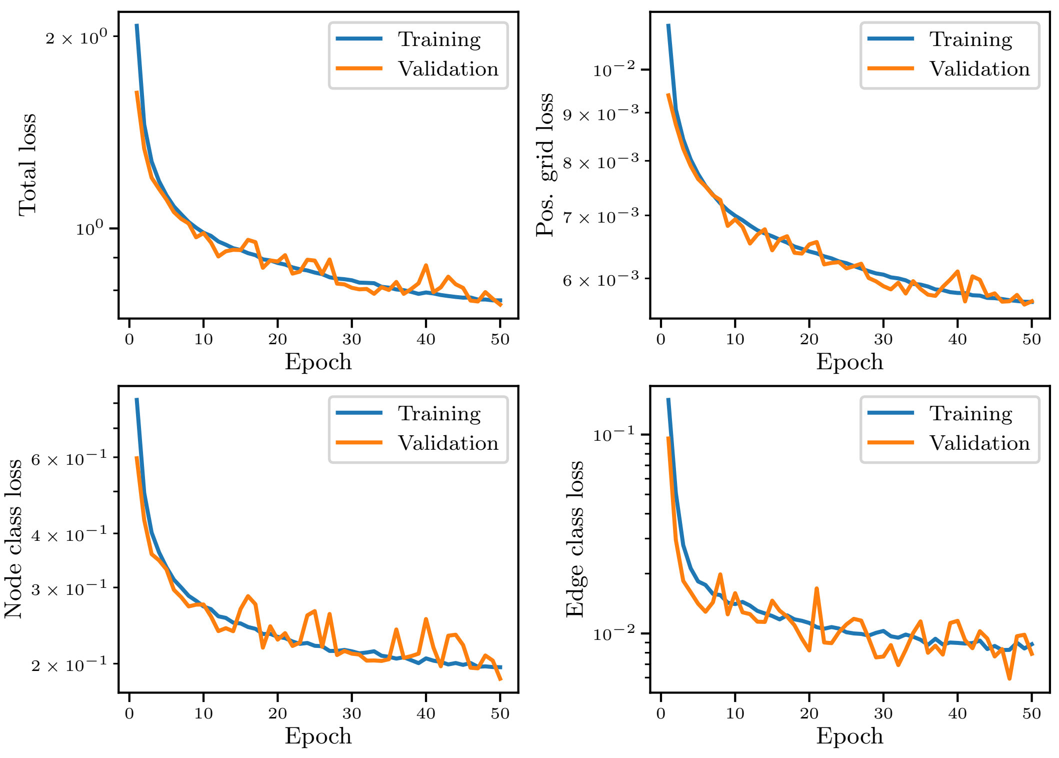

Figure S1: Model training and validation losses as functions of the number of training epochs.

need to often cut out atoms from the molecules, we in general do not have the complete molecule graph available, so we cannot use the SMILES strings. However, the AFM image fixes the spatial ordering of the atoms, so we can use the coordinates of the atoms in order to construct a fixed ordering. There are two major symmetry directions in the in our problem domain. One is the depth-wise direction along the $z$-axis and the second is the lateral direction of the $xy$-plane. The AFM images are equivariant with respect to rotations in the $xy$-plane, meaning that if the molecule is rotated by some angle in the plane, the AFM image remains the same except that it is rotated by the same angle. This is ignoring any effects that the orientation of surface may have, but these effects should be minimal. Therefore, there is no preferred direction in the $xy$-plane. In practice we choose to test an ordering of decreasing $y$-coordinate value, but this choice is in principle arbitrary.

The loss function consist of three parts, the position grid regression error and the classification errors for the node class and the edge connections. The position grid loss is the mean squared error

$$\mathcal{L}_\mathrm{p}(v, \tilde{v}) = \sum_{i=1}^{v_\mathrm{x}} \sum_{j=1}^{v_\mathrm{y}} \sum_{k=1}^{v_\mathrm{z}} |v_{ijk} - \tilde{v}_{ijk}|^2, \tag{1}$$

where $v$ and $\tilde{v}$ are the predicted and reference position grids, respectively. The node classification loss is the categorical cross-entropy

$$\mathcal{L}_{\mathrm{c}}(c', \tilde{c}') = -\sum_{j=1}^{n_{\mathrm{c}}} \tilde{c}'_j \log c'_j, \tag{2}$$

where $c'$ is the SoftMaxed output vector for the node class from the model and $\tilde{c}'$ is the one-hot encoded vector of the correct class. The edge classification loss is the average of the binary cross-entropies for each possible edge connection

$$\mathcal{L}_{\mathrm{e}}(e', \tilde{e}') = -\frac{1}{|V_i|} \sum_{v_k \in V_i} \tilde{e}'_{v_k} \log e'_{v_k} + (1 - \tilde{e}'_{v_k}) \log(1 - e'_{v_k}), \tag{3}$$

where $e'$ is the predicted vector of probabilities for the edge connections and $\tilde{e}' \in \{0, 1\}^{|V_i|}$ is the reference vector of edge connections. On the first iteration ($i = 0$), where $|V_0| = 0$, the loss is instead set to 0. Finally, the total loss is a weighted sum of the above components,

$$\mathcal{L}(v, c', e', \tilde{v}, \tilde{e}', \tilde{c}') = 100\mathcal{L}_{\mathrm{p}}(v, \tilde{v}) + \mathcal{L}_{\mathrm{c}}(c', \tilde{c}') + \mathcal{L}_{\mathrm{e}}(e', \tilde{e}'). \tag{4}$$

The prefactor of 100 for the position grid loss is chosen to roughly balance the loss contributions on the trained model. A comparison of the training and validation losses (Fig. S1) does not show any overfitting during training for any of the loss components.

The predictions are tested by comparing the predicted graphs to the reference graphs. Comparing graph-structured objects is in general a difficult task, and GNN-based models have been developed just for tackling this problem alone [4]. Our case is made easier by the fact that the predicted graph includes coordinates for each node. Comparing the coordinates of the predicted and reference nodes allows us to construct a one-to-one mapping between the nodes in the predicted and reference graphs. Naturally, every node in the predicted graph does not necessarily have a corresponding node in the reference graph, and these nodes are counted separately as missing or extra nodes in the prediction.

The matching is done by first calculating all of the pairwise distances between the nodes in the prediction and the reference. Then for each node in the reference graph, all of the nodes in the predicted graph that are within a chosen threshold distance are gathered. The threshold distance is chosen to be less than half of the smallest possible distance between any two nodes in the reference so that that each node in the prediction is matched to at most one node in the reference. If there are no matches within the threshold distance, that node is counted as a missing node, and for one or more matches, the closest node is chosen for the mapping. Any nodes in the prediction that were not mapped to any node in the reference are counted as extra nodes.

Once the nodes have been mapped, the predicted and reference classes for the nodes can be gathered into a confusion matrix where each entry is defined as

$$M_{ij} = \text{number of nodes with reference class } i \text{ and predicted class } j. \tag{5}$$

We can further define the precision and recall for each class $k$:

$$\text{Precision}(k) = \frac{M_{kk}}{\sum_{i=1}^{n_c} M_{ik}} \tag{6}$$

$$\text{Recall}(k) = \frac{M_{kk}}{\sum_{j=1}^{n_c} M_{kj}}. \tag{7}$$

The precision for a class is a measure of the confidence that a prediction in that class is actually of that class, and the recall tells what fraction of all cases in that class are found by the predictions. The confusion matrix and the precision and recall are defined similarly for the edge connections where the two classes are the presence or absence of an edge between a pair of nodes.

## References

(1) Oinonen, N.; Xu, C.; Alldritt, B.; Canova, F. F.; Urtev, F.; Cai, S.; Krejčí, O.; Kannala, J.; Liljeroth, P.; Foster, A. S. Electrostatic Discovery Atomic Force Microscopy. *ACS Nano* **2022**, *16*, 89–97.

(2) Williams, R. J.; Zipser, D. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Comput.* **1989**, *1*, 270–280.

(3) Li, Y.; Vinyals, O.; Dyer, C.; Pascanu, R.; Battaglia, P. W. Learning Deep Generative Models of Graphs. *CoRR* **2018**, *abs/1803.03324*.

(4) Li, Y.; Gu, C.; Dullien, T.; Vinyals, O.; Kohli, P. Graph Matching Networks for Learning the Similarity of Graph Structured Objects. *CoRR* **2019**, *abs/1904.12787*.

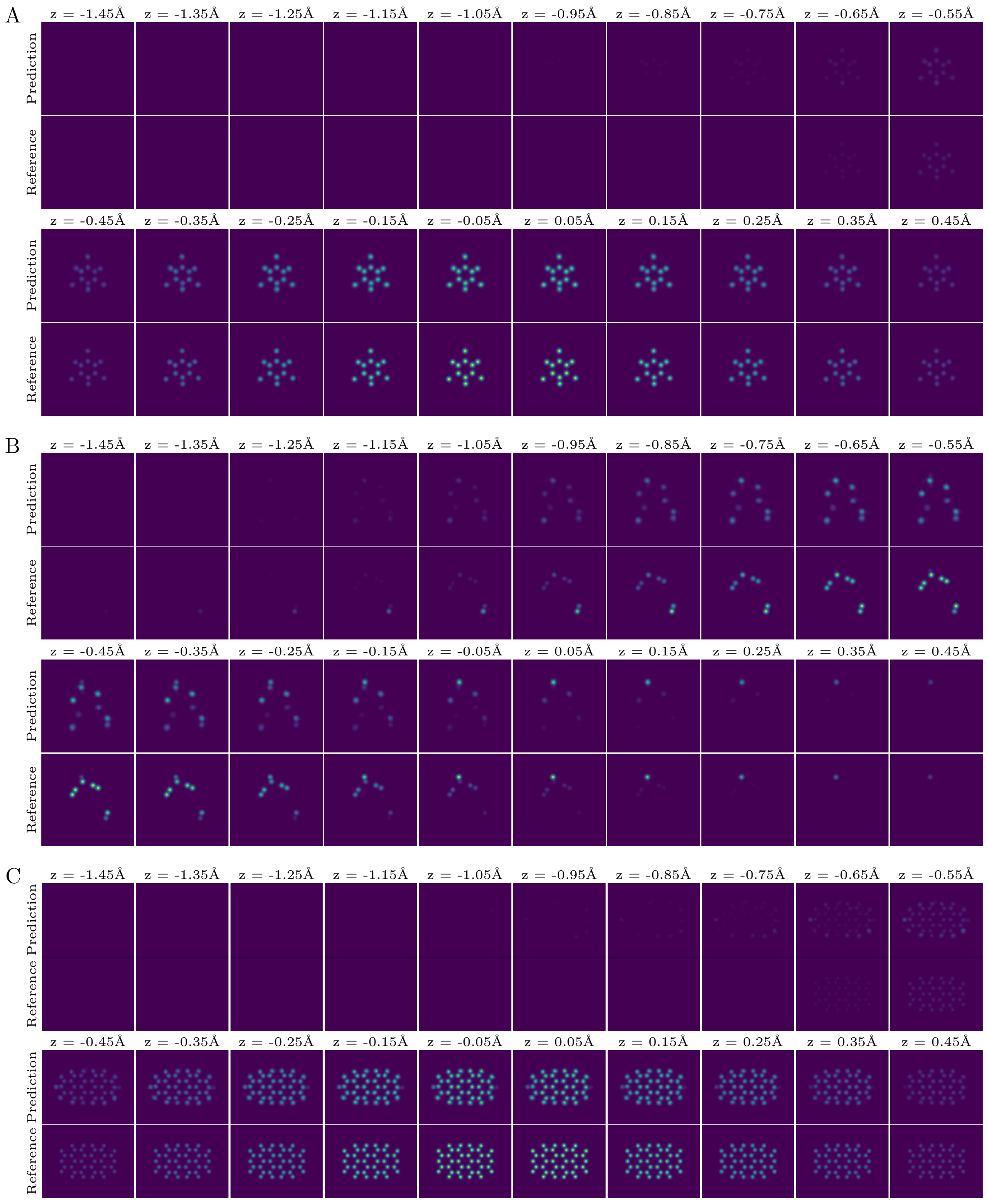

Figure S2: Full 3D position grids for the predictions in Fig. 2 of the main article.

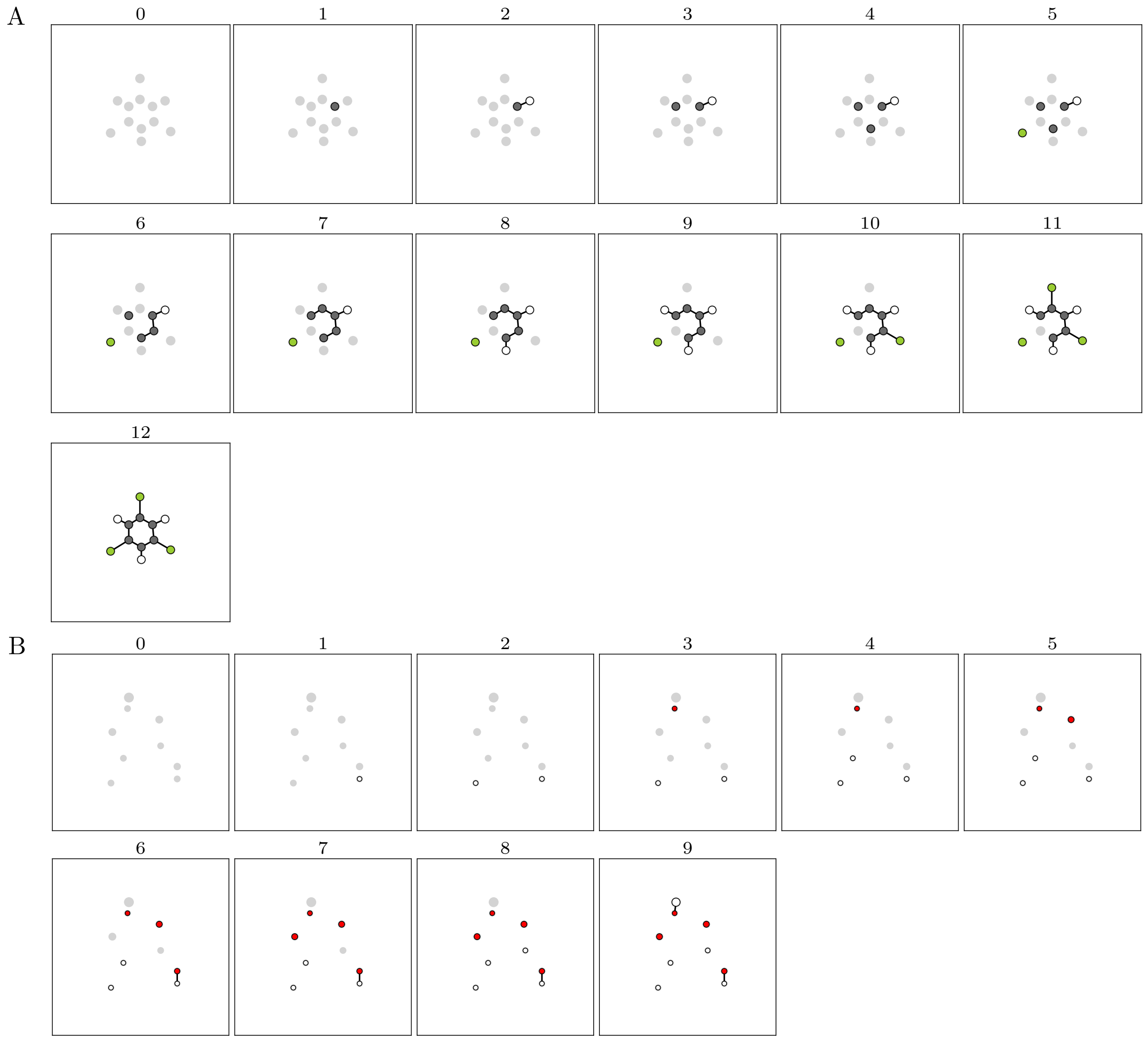

Figure S3: Graph construction sequences for the three example systems in Fig. 2 of the main article. Light-grey circles without outline correspond to positions of atoms that have not yet been added to the graph. Colors of the circles correspond to the atom classes as in Fig. 2.

C

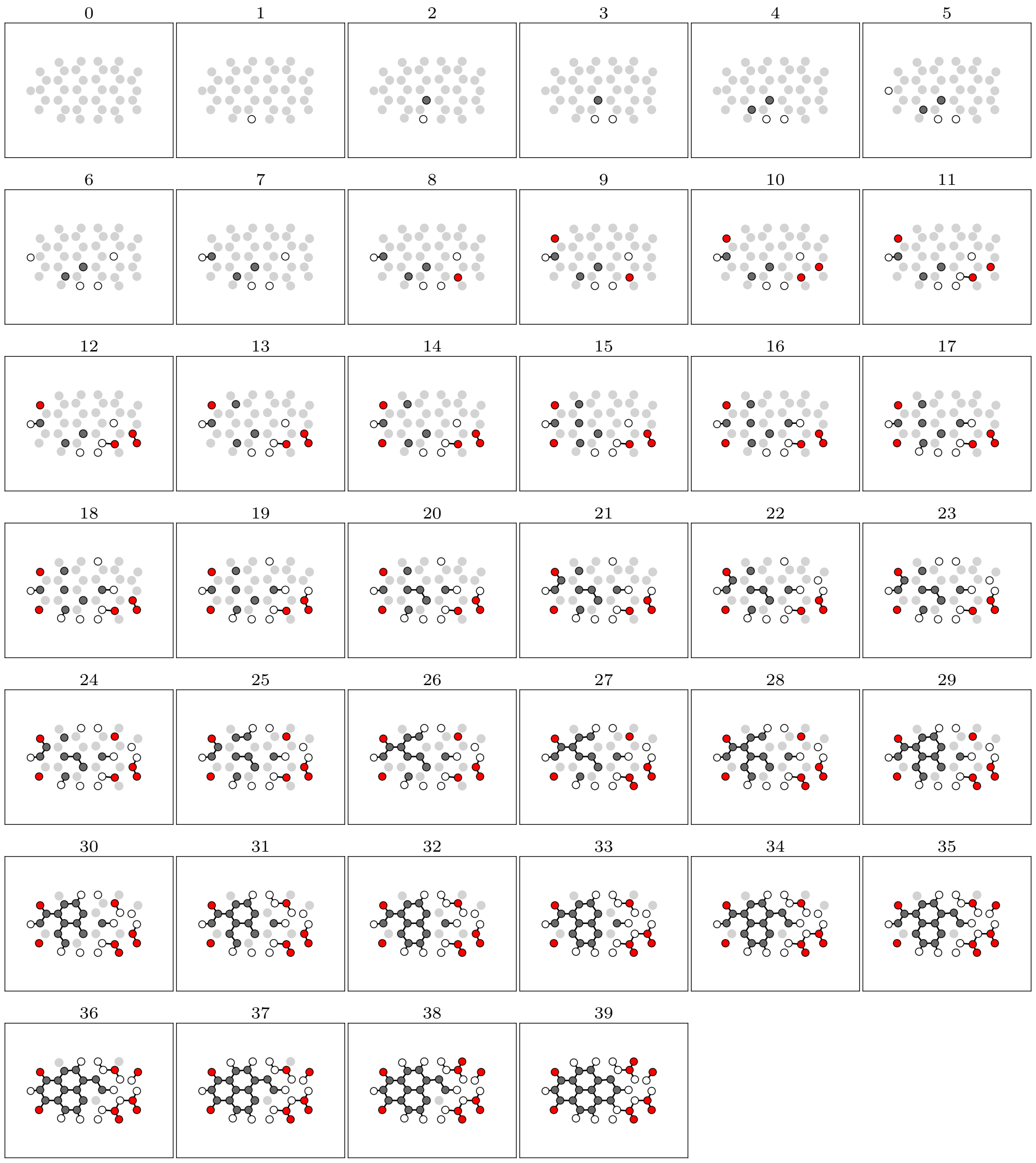

Figure S3: (Continued) Graph construction sequences for the three example systems in Fig. 2 of the main article. Light-grey circles without outline correspond to positions of atoms that have not yet been added to the graph. Colors of the circles correspond to the atom classes as in Fig. 2.

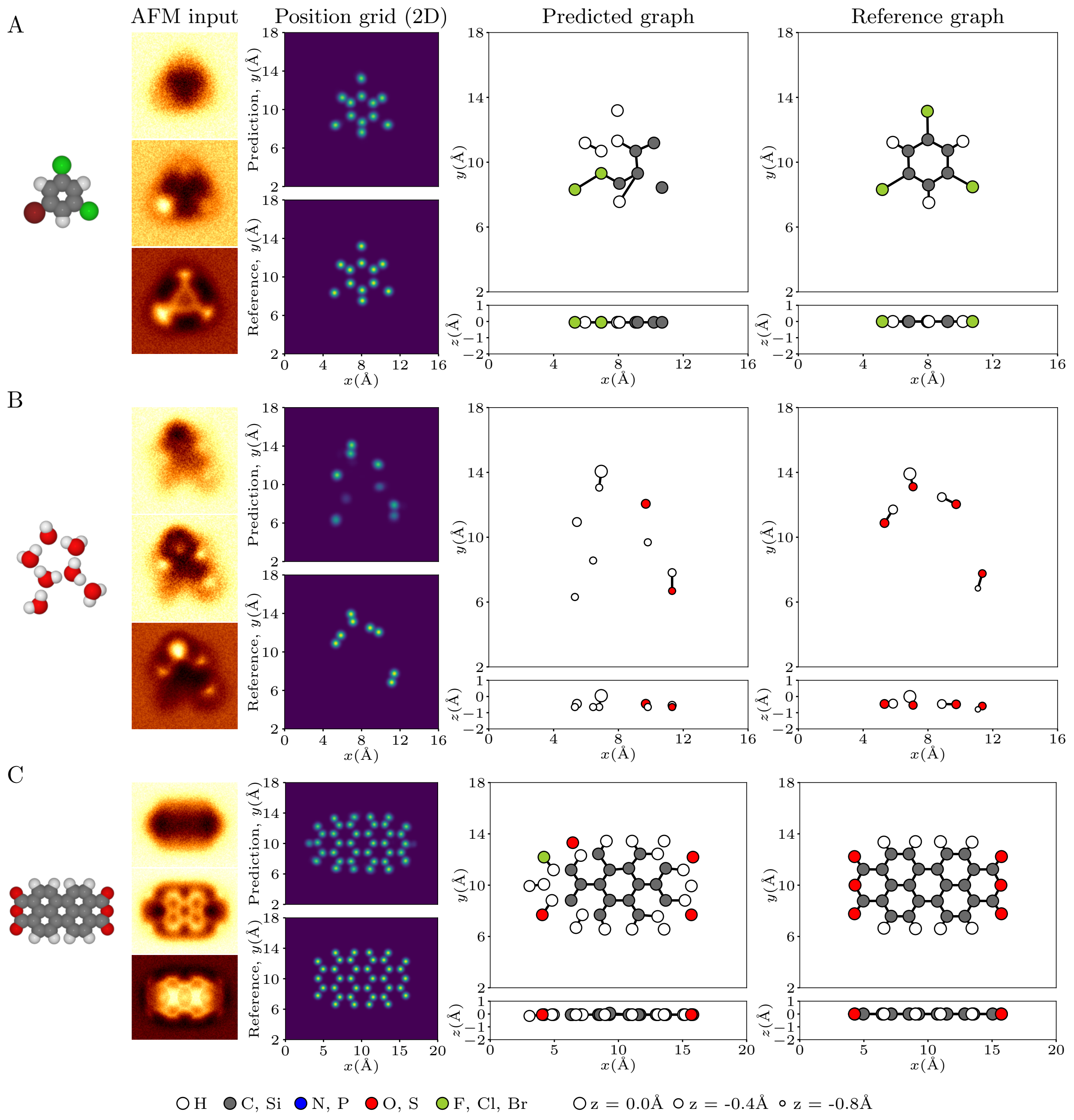

Figure S4: Graph predictions on the three test systems with coordinates shifted by $-2$ Å in the x direction.

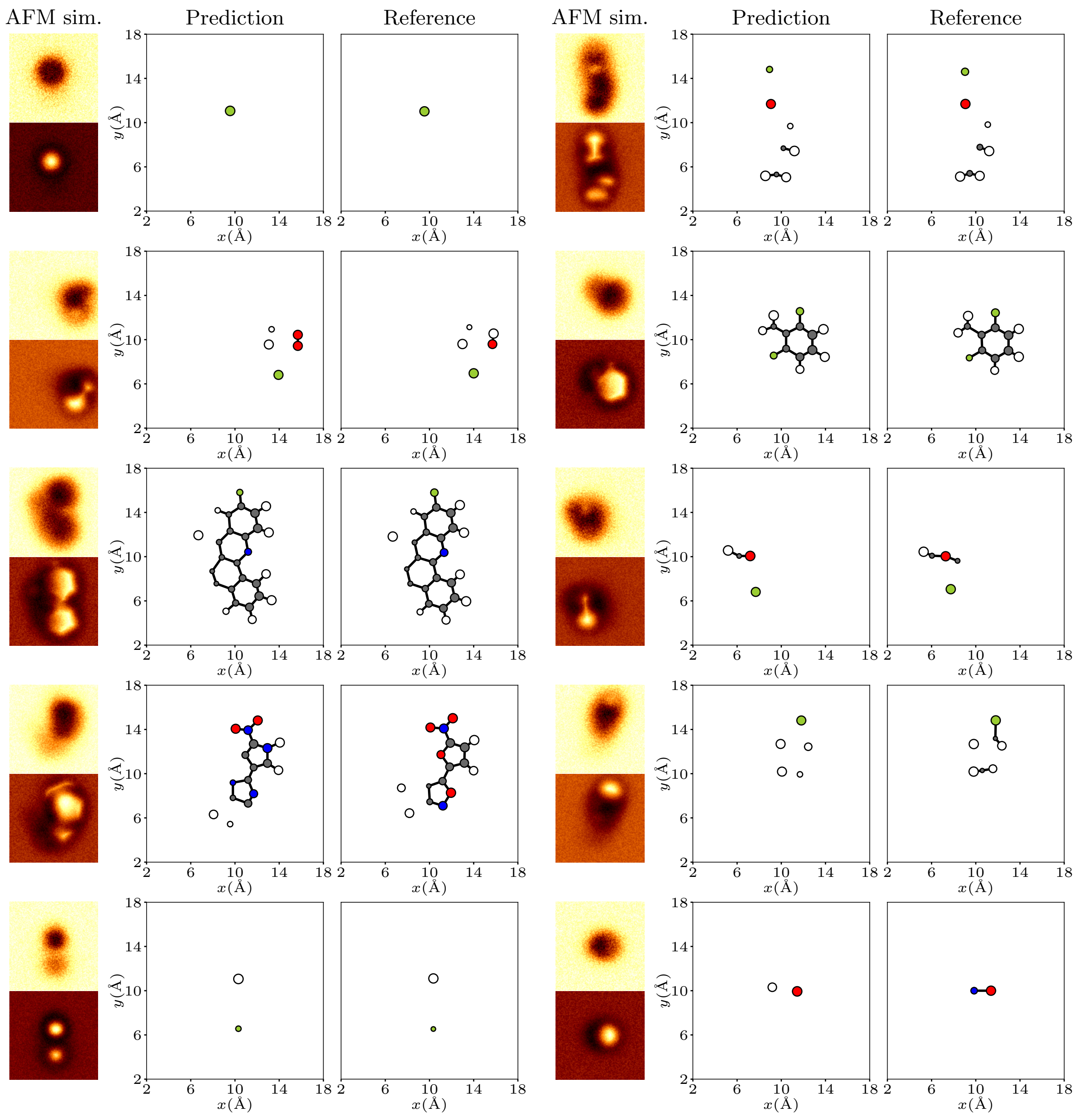

Figure S5: Additional predictions on randomly chosen samples from the test set. For each sample, on the left are shown the furthest and closest slice of the simulated input AFM image set and on the right are shown the prediction and the reference in the xy-plane projection.

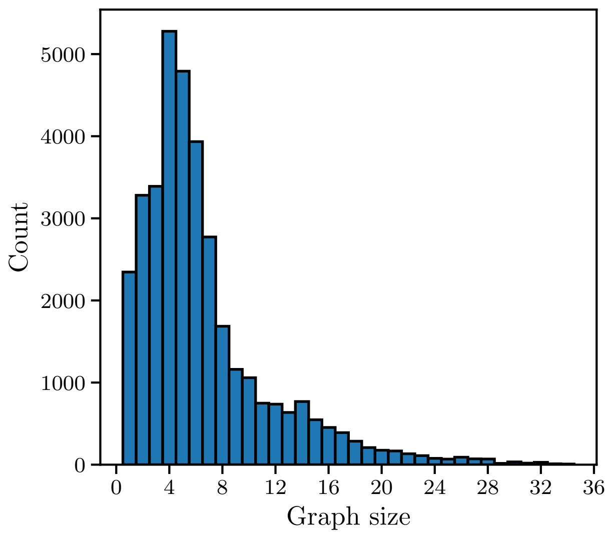

Figure S6: Histogram of graph sizes in the test set.

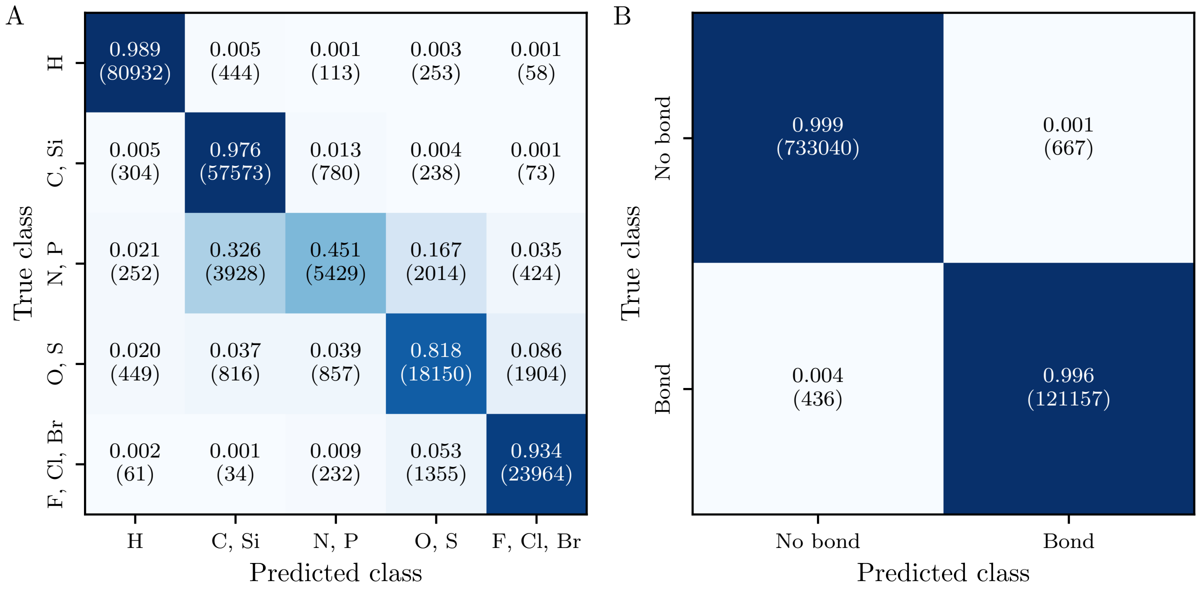

Figure S7: Confusion matrices of (A) atom classification and (B) edge classification for y-order graph construction.

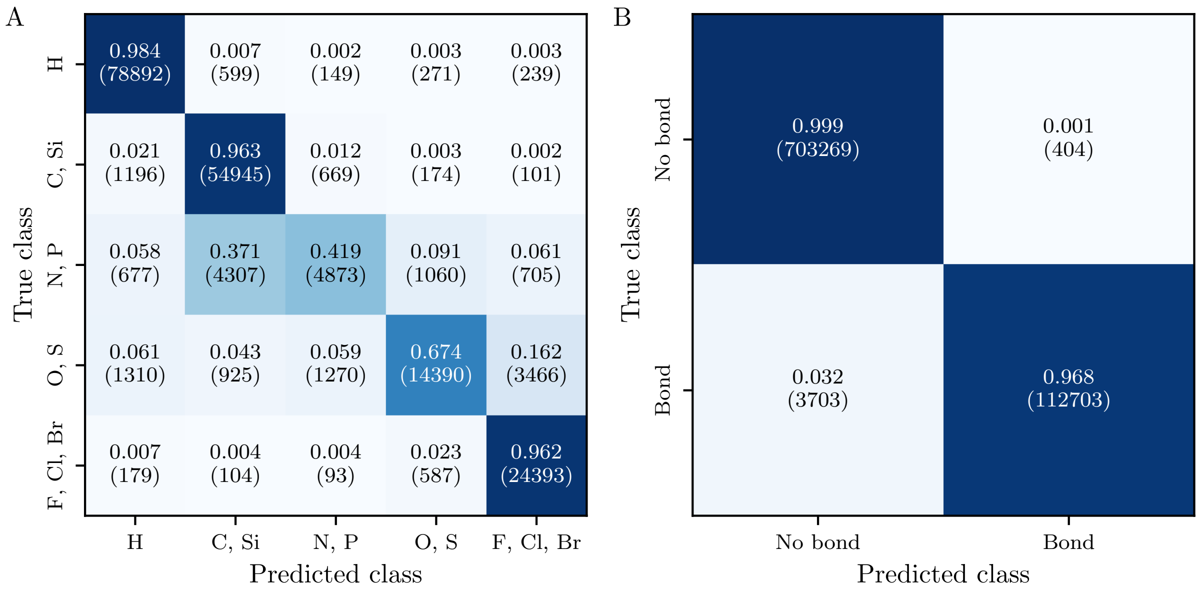

Figure S8: Confusion matrices of (A) atom classification and (B) edge classification for z-order graph construction.

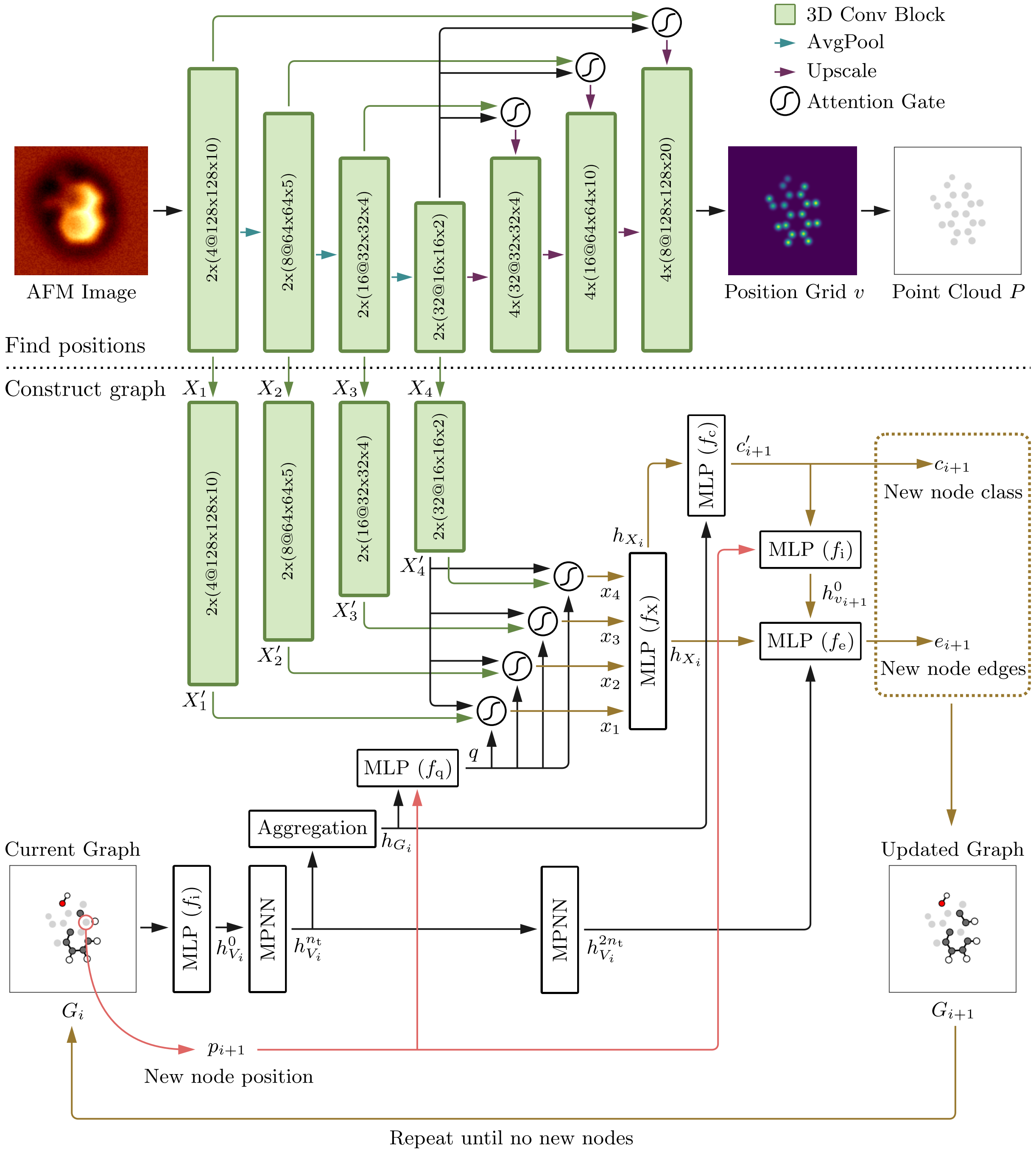

Figure S9: Schematic of information flow inside the model. In the 3D convolution blocks, the size of the feature maps is presented for an AFM images with lateral input size of $128 \times 128$, and the first number indicates the number of layers in the block. For explanations of symbols, see the main text.