Supplementary Information for:

DeepStruc: Towards structure solution from pair distribution function data using deep generative models

Emil T. S. Kjær[†1], Andy S. Anker[†1], Marcus N. Weng[1], Simon J. L. Billinge[*2,3], Raghavendra Selvan[*4,5], Kirsten M. Ø. Jensen[*1]

†Both authors contributed equally to this work.

*Correspondence to sb2896@columbia.edu, (SJLB), raghav@di.ku.dk (RS), kirsten@chem.ku.dk (KMØJ)

1: Department of Chemistry and Nano-Science Center, University of Copenhagen, 2100 Copenhagen Ø, Denmark

2: Department of Applied Physics and Applied Mathematics Science, Columbia University, New York, NY 10027, USA

3: Condensed Matter Physics and Materials Science Department, Brookhaven National Laboratory, Upton, NY 11973, USA

4: Department of Computer Science, University of Copenhagen, 2100 Copenhagen Ø, Denmark

5: Department of Neuroscience, University of Copenhagen, 2200, Copenhagen N

Table of Contents

A: Distribution of the seven structure types in data for mono-metallic nanoparticle (MMNP) structure solution

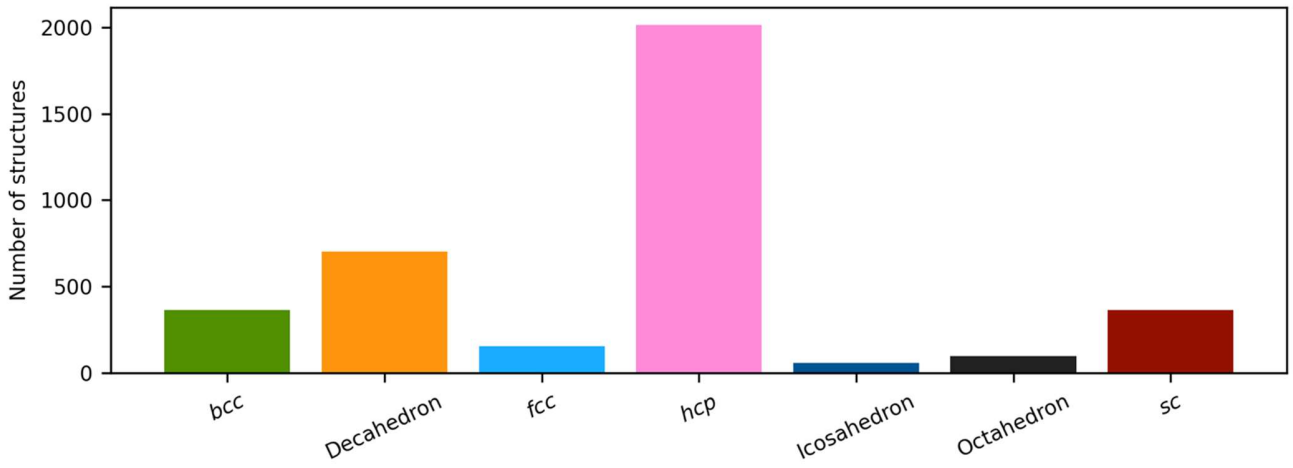

Fig. S1 | The distribution of the seven structure types in the dataset used for MMNP structure solution. In total, the dataset consists of 3742 structures, whereof 361 are body-centered cubic (*bcc*), 703 are decahedral, 152 are face-centered cubic (*fcc*), 2014 are hexagonal closed packed (*hcp*), 57 are icosahedral, 95 are octahedral and 361 are simple cubic (*sc*).

B: Method

In the following sections, we briefly explain what a PDF is, how we obtained the simulated PDFs and their structures, and finally we elaborate on the CVAE method developed here to analyse PDFs. A more detailed description of the PDF is given elsewhere.[1]

B.1.: The Pair Distribution Function (PDF)

The PDF is the Fourier transform of total scattering data, which can be obtained through x-ray, neutron, or electron scattering. In this work we focus on the usage of x-ray total scattering data. The scattering vector Q is defined as follows, where λ is the radiation wavelength, and θ is the scattering angle:

$$Q = \frac{4\pi sin(\theta)}{\lambda}$$

The measured scattering intensities are denoted I(Q), which are corrected for incoherent scattering, fluorescence, etc. and normalized such that the total scattering structure function S(Q) is obtained.

$$S(Q) = \frac{I(Q) - \langle f(Q)^2 \rangle + \langle f(Q) \rangle^2}{\langle f(Q) \rangle^2}$$

Here f is the atomic form factor. To obtain the structural real-space information, the total scattering structure function is Fourier transformed over the truncated Q-range, hence yielding the reduced PDF also known as G(r):

$$G(r) = 2/\pi \int_{Q_{min}}^{Q_{max}} Q[S(Q) - 1] \, sin(Q \cdot r) dQ$$

G(r) can be interpreted as a histogram of real-space interatomic distances and the information is equivalent to that of an unassigned distance matrix (uDM). Simulated PDFs are shown in Fig. 1b and all simulation parameters can be found in section G in the Supplementary Information. The PDFs used in this project are normalised to have I(G(r)) = 1 as illustrated in section H in the Supplementary Information.

B.2.: Simulated and experimental data

To simulate the nanoparticles used in the training process of DeepStruc, the Python library atomic simulation environment (ASE) was used.[2] The seven different structure types: *fcc*, *bcc*, *sc*, *hcp*, icosahedral, decahedral, and octahedral were constructed with the cluster module in ASE in the same manner as described by Banerjee et al.[3] and Anker & Kjær et al.[4] All MMNPs were generated in sizes ranging from 5 to 200 atoms. Each MMNP

was then populated with different atoms hence changing the lattice spacing/bond distances in the MMNP. To ensure that there were no duplicate MMNPs within the dataset, all MMNPs were decomposed into a distance list of all atom-atom distances. The distance lists are a reduced format of the xyz representation as they are rotation- and translation-invariant in Euclidean space. All the distance-lists were sorted and duplicate structures with equivalent distance lists were removed. This yielded a total of 3742 unique MMNPs, see section A in the Supplementary Information for the distribution of the seven structure types. The xyz-coordinates will be the label that DeepStruc has to reconstruct. Nanoparticles with each of the seven structure types can be seen in Fig. 1b along with their simulated PDF, Fig. 1a. All the simulation parameters used can been seen in section G in the Supplementary Information.

To further investigate the latent space behaviour of DeepStruc, a more chemically simple and intuitive dataset was made of *fcc*, *hcp,* and stacking faulted structures. *Fcc* and *hcp* can be considered layered structures that are only differentiated by the repetition of layers within the structure. *Fcc* consists of a repeated ABCABC layered structure where *hcp* is an ABABAB layered structure. A 5 layered stacking fault structure could then be described as ABCAC, as it does not satisfy either of the *fcc* or *hcp* stacking criteria, see Fig. 6. A total of 1620 stacking fault structures were generated.

B.3.: Data representation

In this work, the structures from ASE are converted into a graph-based representation in order to capture the interatomic relationships, as the original representation generated with ASE are not optimal as input to DeepStruc. Graph representations have seen increasing success in machine learning applications related to materials science as the interatomic relations in graphs are invariant to transformations of the structure such as solid translations and rotations.[5,6] Each structure in graph representation can be described as G = (**X**,**A**), where $\mathbf{X} \in \mathbb{R}^{N \times F}$ is the node feature matrix which contains F features that can describe each of the N atoms in

the structure. We use F = 3 comprising only the Euclidean coordinates of the atom in a 3-dimensional space. The interatomic relationships are captured using the adjacency matrix $\mathbf{A} \in R^{N\times N}$. In our case, the entries of the adjacency matrix are the Euclidean distance between each pair of atoms, resulting in a soft adjacency matrix. However, to make the adjacency matrix sparse, when the distance between any pair of nodes is larger than the lattice constant the corresponding edge weight is set to zero. When the edge weight is zero this corresponds to absence of an edge between the pair of nodes, and in other cases the edges have a weight given by the interatomic distance. Section I in the Supplementary Information shows a decahedron consisting of seven atoms alongside the components describing it in our chosen graph representation.

B.4.: The Conditional Deep Generative Model (DGM)

DGMs such as variational autoencoders (VAEs) are commonly used to synthesize novel, synthetic data by approximating the underlying data-generating processes based on the training data.[7] In this work, we are interested in generating structures based on properties such as the PDF resulting in the conditional DGM scenario. The specific formulation of the conditional DGM used in this work is the CVAE, initially proposed for computer vision tasks[8] and more recently it has also been explored for synthesizing novel drug molecules.[9] The CVAE in this work is trained to solve the unassigned distance geometry problem[10] (uDGP) as it solves the task of converting the distances within a PDF to a chemical structure. In the uDGP the problem of taking a starting point of a list of distances and reconstructing it into a structure is broken down into two discrete problems. First, is to discover the graph that connects pairs of atoms, with the edges labelled by the distances from the distance list (the assignment problem). Second is to embed this graph into Euclidian space. An illustration of the CVAE can be seen in Fig. 1a. Here, the blue area is the training process, and the green area is the prediction/inference process. During training of the CVAE, the encoder takes pairs of structures and their corresponding PDFs as input. The encoder learns to map the structure-PDF pairs into a low-dimensional,

latent Gaussian distribution, known as the encoder distribution. Each structure-PDF pair is mapped to certain regions of the latent space. When trained with large amounts of diverse data, the latent space is able to capture relationships between different structures and PDF pairs so that similar structures are closer in this latent space than very different structures. CVAEs are different from classical autoencoders in that the latent space is probabilistic, which makes it possible to sample structures from these latent encoder distributions. This is achieved during training by forcing the encoder distributions to align with a simpler prior distribution which only takes the PDF as input. The two distributions are matched by minimizing the Kullback-Leibler Divergence between the encoder and prior distributions and is interpreted as the regularization term, L_{reg}.

The prior NN gets the PDF as input and maps it to the low-dimensional prior distribution. The low-dimensional latent vector conditioned on the PDF is then input to the decoder, which is tasked to predict the xyz-coordinates of the structural input. During the training process, the mean squared error (MSE) between the xyz-coordinates of the input and output are computed to force the decoder to predict xyx-coordinates from the latent representations. The MSE is defined as the reconstruction loss, L_{rec}. The CVAE is trained by jointly optimizing these two loss components:

$$L_{CVAE} = L_{rec} + \beta \cdot L_{reg}$$

where β is a scaling factor that controls the relative influence of the regularization- and reconstruction-terms. In our training process, at initialization β is set to 0 which allows the model to focus on minimizing L_{rec}. Each time L_{rec} gets below a certain threshold β is increased. This helps keep the model from falling into a local minimum and the process is repeated until convergence has been reached. Similar strategies for annealing β in VAEs have been attempted.[11,12] At inference (test) time, the prior NN receives the PDF as input which is then mapped to the low-dimensional latent space which during training has been trained to match the encoder distribution. A sufficiently well trained CVAE is then able to predict structures from the latent space based on

the PDF input. A simplified version of the CVAE used for this work, DeepStruc, can be seen in Fig. 1a. The CVAE is presented more formally in the work by Anker and Kjær et al...[4]

B.5.: Graph Conditional Variational Autoencoder (CVAE)

In this work, two types of CVAEs were utilized depending on the type of encoder. In the conventional CVAE, the encoder was based on Multi-Layered Perceptrons which operate on a tabular format of the node features, and the adjacency matrix populated with atom–atom distances. For the second type of CVAE – that we call the graph CVAE – the encoder consists of a graph neural network (GNN)[6,13] and is able to process graph structured data, taking the neighbourhood information into consideration. GNNs are generalized message passing methods that can aggregate information from the neighbourhood of a node by passing messages along the edges. These messages are learned during training and can summarize the information present at the node necessary for the downstream tasks. Further, by making the encoder deep, i.e. adding additional GNN layers, nodes can get access to information from nodes that are farther from them. For instance, in a k-layered GNN each node had access to information from nodes that are k-hops away. In our experiments, we observed that the generative capabilities of the graph CVAE was better than the conventional CVAE, part E in the Supplementary Information. Further, we were able to obtain comparable reconstruction quality from the graph CVAE with only two latent dimensions compared to using eight dimensions for the conventional CVAE. This indicates that the graph encoder is able to better compress the information present in the node and adjacency matrices. A minor technical detail in our CVAE models is that the predictions from the decoder do not exactly match the input features. That is, the decoder does not reconstruct the full input comprising node features and adjacency matrix but only the node features. The algorithm we refer to as DeepStruc is a graph based CVAE.

C: Fitting parameters and mean absolute error (MAE) measures of reconstructed MMNPs for analysis of simulated pair distribution functions (PDFs)

In the PDF comparisons a scale-factor, a contraction/expansion-factor and an isotropic atomic displacement parameter (ADP) are refined. This approach is taken in brute force methods such as clusterMining[3] for deciding on the best cluster for a given measured PDF. It accounts for small differences in bond-length, scale-factor, and thermal motion without changing the geometric arrangements of the atoms in the clusters

Table S1 | Refined parameters for MMNPs obtained from the DeepStruc algorithm for analysis of simulated PDFs. The MAEs for the predicted xyz-coordinates are given along with the R_{wp}-values obtained in the PDF fit. The structures are shown in Fig. 3 of the main paper.

Name	#Atoms	Scale	Expansion/ contraction factor	B_{iso}	MAE [Å]	MAE fit [Å]	R_{wp} [%]
bcc	89	0.146	0.959	0.428	0.132 ± 0.086	0.083 ± 0.068	21.2
Decahedral	105	0.107	1.030	0.238	0.140 ± 0.095	0.116 ± 0.081	39.0
fcc	171	0.117	0.957	0.284	0.182 ± 0.073	0.116 ± 0.053	54.2
hcp	128	0.082	0.974	0.081	0.093 ± 0.041	0.043 ± 0.025	10.8
Icosahedral	55	0.119	0.962	0.390	0.120 ± 0.058	0.092 ± 0.044	37.8
Octahedral	146	0.111	0.975	0.306	0.132 ± 0.056	0.106 ± 0.051	48.2
sc	177	0.158	0.984	0.210	0.091 ± 0.045	0.091 ± 0.043	46.4

D: Implementation of the sampling and fitting process of the predicted structures from latent space

During the inference process, a PDF is embedded into the latent space using a normal distribution by the prior neural network. Samples drawn from this normal distribution can be used to obtain predicted structures to be fit to the PDF. For the simulated test set (seven structure types and stacking faulted) only a single sample was drawn

from the latent space to obtain structures for each of the PDFs. For the experimental data, in order to explore the latent space to a larger extent, we scale the standard deviation (σ) of the normal distribution by σ: 3, 5 and 7. We then chose to sample 1000 predicted structures for each of the three normal distributions (σ: 3, 5 and 7) and fit them to the data. For the experimental data we report the best fitted structure within each of the distributions (Supplementary Information section E). Fig. S3 demonstrates the histogram of x- and y-positions in latent space when sampling from a mean latent space position of (9.7, 14.8) using the σ: 3 distribution for the $Au_{144}(PET)_{60}$. We here sample from a normal distribution, however this could be optimized in many ways e.g. by drawing structures from a uniform distribution, by sampling from latent space regions that are well-populated by the test set, by favouring sampling from regions in the latent space that is populated by underrepresented structures, or by introducing chemical knowledge into the sampling process. Another way to alter the latent space would be to deviate from the isotropic normal distributions. The latent distribution of the CVAE is constrained to be a symmetric normal distribution (with a diagonal covariance matrix). In some cases, this might be limiting and one could consider allowing additional flexibility to the latent space structure. One of the ways of doing this is by relaxing the diagonal covariance matrix assumption, so that asymmetric normal distributions can be modelled. This is previously studied by Jakub et al.[15] where more complex structures for the covariance matrix are modelled for VAE-type models. This additional flexibility of the latent distribution increases the complexity of the learning problem but could also provide efficient exploration of the latent space.

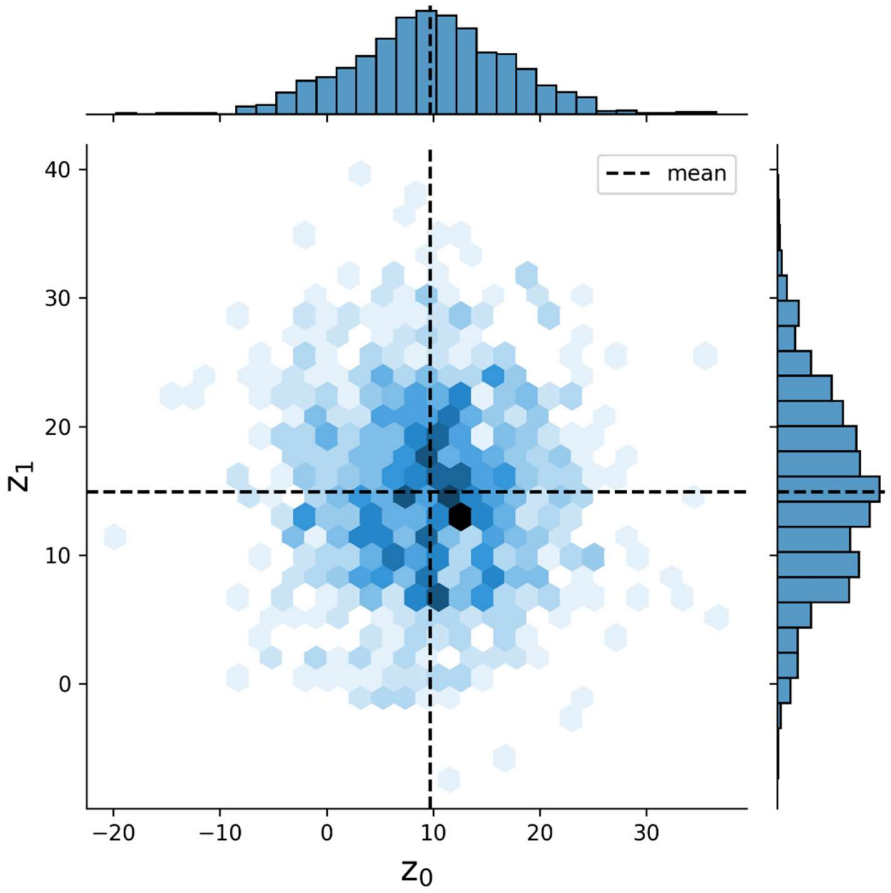

Fig. S3 | Representative histogram of z_0- and z_1-positions in latent space when sampling from a mean latent space position of (9.7, 14.8) using the σ: 3 distribution for $Au_{144}(PET)_{60}$. The black dashed line indicates the mean sampling position.

E: Comparing the DeepStruc with baseline algorithms

To evaluate the results, we compare our results with two different approaches that can be used to identify the structural model.

The first approach is the brute-force approach, further described in the part E.1, proposed by Banerjee et. al.,[3] which is fitting all constructed MMNP structures to the dataset and reporting the R_{wp} value. The second approach is a tree-based supervised learning algorithm, further described in part E.2, which has been trained on the same

MMNP structures set as DeepStruc but using 100 PDFs with different simulation parameters for each MMNP structure. The range of simulation parameters used is shown in Table S5. While the brute-force method is directly providing us with the best fit, it is computationally expensive. The tree-based algorithm can, like DeepStruc, predict the chemical structure based on its PDF in less than a second. However, the brute-force approach and the tree-based algorithm cannot map a low-dimensional space of chemical structures that can be used to analyse similarities between structures. Also, they are constrained to the structural database whereas the regular CVAE without a graph-based input and the DeepStruc algorithm has generative capabilities which make it possible to both interpolate and extrapolate slightly from the training distribution. Fig. S2 illustrates a comparison of the results of the brute-force, tree-based and DeepStruc. The fits from the brute-force approach have a slightly lower R_{wp} value than the fits from the tree-based approach and DeepStruc. However, the brute-force approach is at least 3 orders of magnitude slower and consumes at least 4 orders of magnitudes more CO_2 than the Machine Learning (ML) algorithms after they have been trained, see section E.3.. The training process of the ML algorithms has to be done only once compared to the brute-force algorithm which has to be redone on every experimental dataset that is collected.

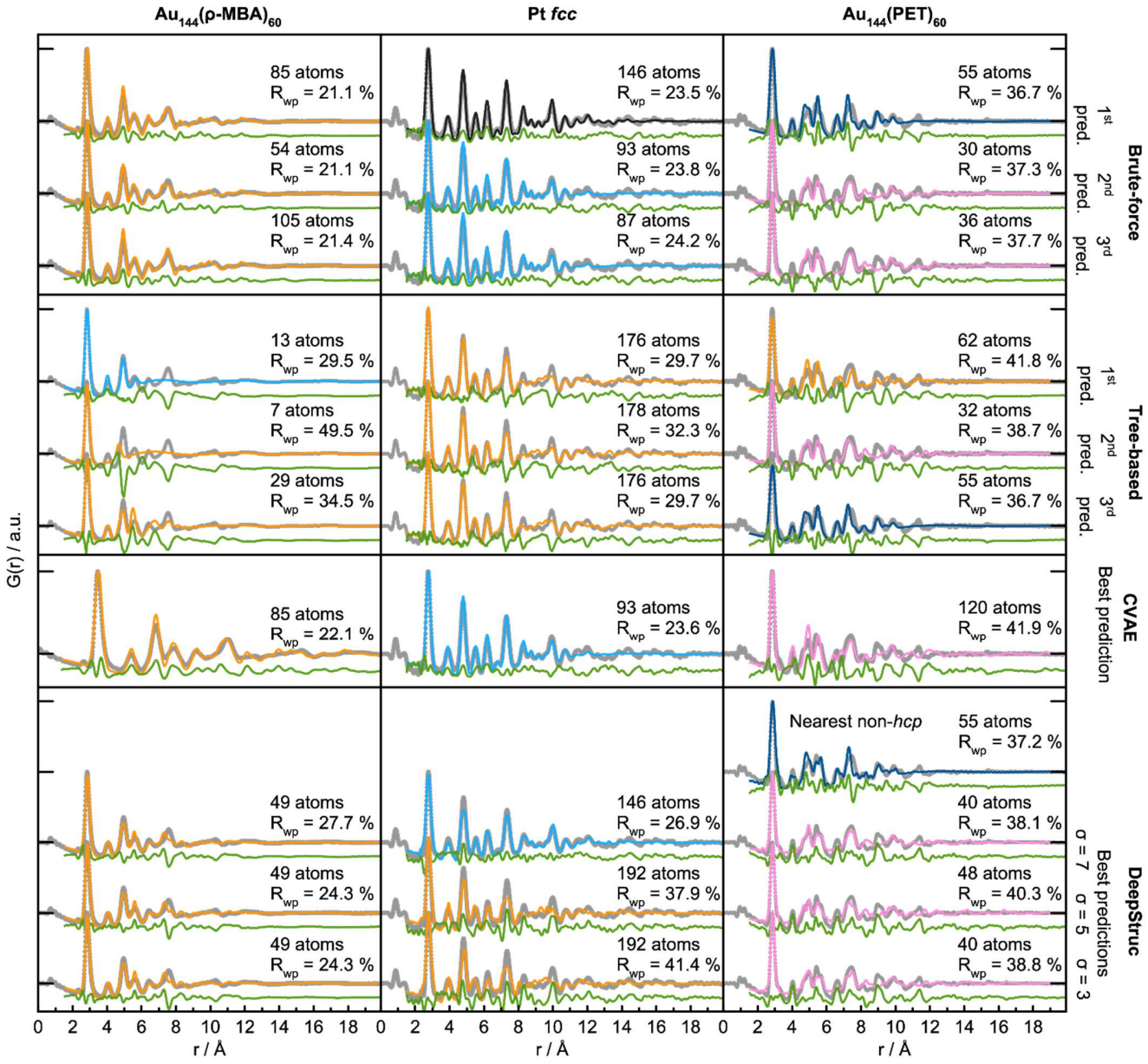

Fig. S4 | Comparing DeepStruc with baseline models. DeepStruc are used to predict a structure from each of the three different datasets, Au_{144}(*p*-MBA)$_{60}$, the Pt *fcc* and $Au_{144}(PET)_{60}$[14,16] as described in section D. The structures predicted by DeepStruc are compared to the top-3 predictions from the brute-force approach and the tree-based model. The fits have been made using a scaling factor, an isotropic expansion of the structure and an isotropic atomic displacement parameter (ADP) in DiffPy-CMI.[17]

E.1.: Brute-force modelling

Brute-force metal mining was done by creating MMNPs with the Python library atomic simulation environment (ASE) module[2] and fitting them to the data iteratively in DiffPy-CMI[17], from 0 to 30 Å, as proposed by Banerjee et al.[3] The advantage of the brute-force approach is that it directly yields the R_{wp} value of the fit of the structure to the dataset. The disadvantage is that it is computationally expensive. In this project, the results of the brute-force approach are used as a baseline for the predictions from the ML predictions.

E.2.: Using a tree-based classification algorithm to predict a MMNP from a PDF

A gradient boosting decision tree (GBDT) algorithm was trained to do the classification job of predicting the MMNPs from a PDF.[18] For each MMNP, 130 PDFs were simulated with parameters in the range shown in Table S5. The GBDT algorithm is trained on 100 of the PDFs, 15 of the PDFs were used for validating the model during the training process and 15 of the PDFs were used to calculate the accuracy of the model after the training process (test set). The model is trained with a learning rate on 0.15, max depth on 3 and an early stopping criteria on 5 rounds of no improvement.

Table S2 | DiffPy-CMI simulation parameters for PDF data of the data used to train the tree-based classifier.

r_{min} (Å)	0.0	**Q_{min} (Å^{-1})**	0–2	**B_{iso} (Å^{-2})**	0.1–1
r_{max} (Å)	30.0	**Q_{max} (Å^{-1})**	12–25	**Delta2 (Å^{-2})**	2.0
r_{step} (Å)	0.1	**Q_{damp} (Å^{-1})**	0.01–0.04		

The loss curve, Fig. S5, illustrates that the GBDT did not improve significantly after about 100 epochs.

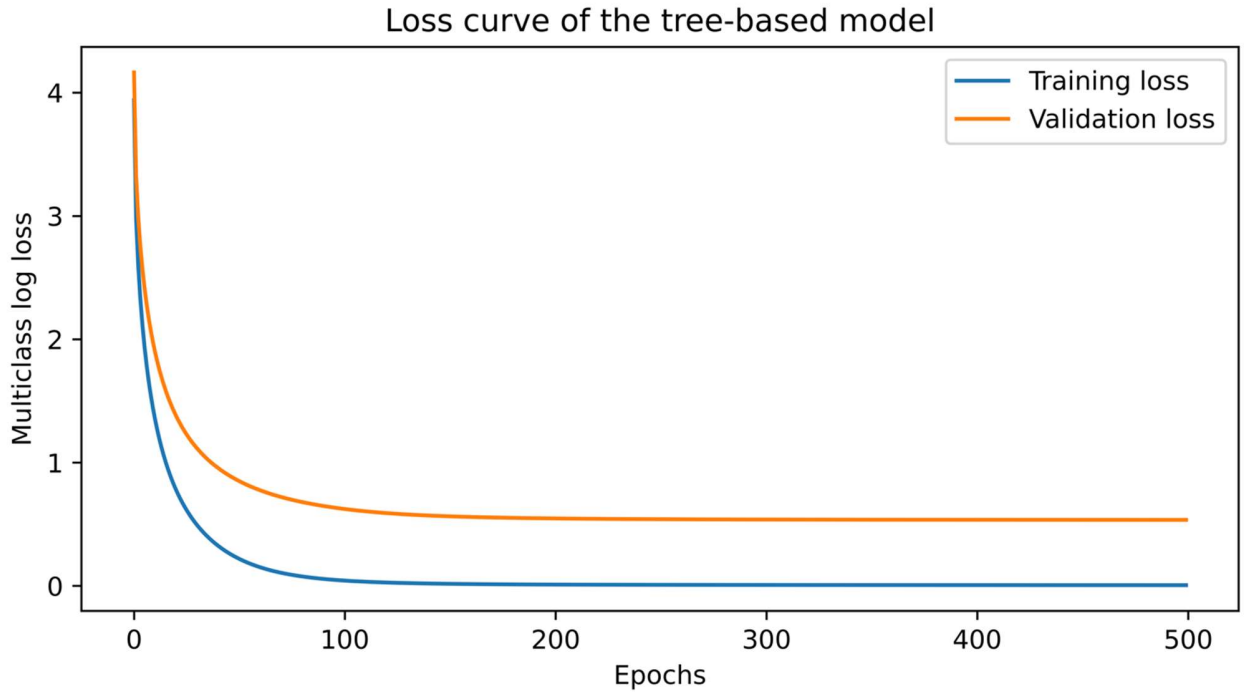

Fig. S5 | Loss curve from the training process of the GBDT.

The model subsequently predicts with an 83.87 % accuracy on the test set. Accuracy from top 3 is 95.95 % and accuracy for top 5 is 97.89 %. The GBDT algorithm does often yield very similar results to the brute-force modelling method, despite for $Au_{144}(PET)_{60}$, where it does not identify the icosahedral structure as the best fitting structure. We believe this is because the icosahedral structure was underrepresented in the model, which can be fixed with data augmentation.

E.3.: Tracking the CO_2 emission of DeepStruc and the baseline models using CarbonTracker[19]

Table S3 | CO_2 cost due to the energy consumption of the brute-force approach, the tree-based classifier and DeepStruc to determine the structure of the three experimental files - $Au_{144}(PET)_{60}$, $Au_{144}(p\text{-MBA})_{60}$ and the Pt *fcc* MMNP.[14,16] Generating the MMNPs takes about 2 hours and 53 min (0.12 kWh).

Method	Training process (kWh / min)	Prediction process (Wh / sec)
Brute-Force	-	250 / 51840
Tree-based Classifier	0.76 / 1229	0 / 0
DeepStruc	3.81 / 872	0 / 0
DeepStruc (stacking fault)	1.86 / 514	0 / 0

F: The Pt *fcc* Pair Distribution Functions (PDFs) from Quinson et al.

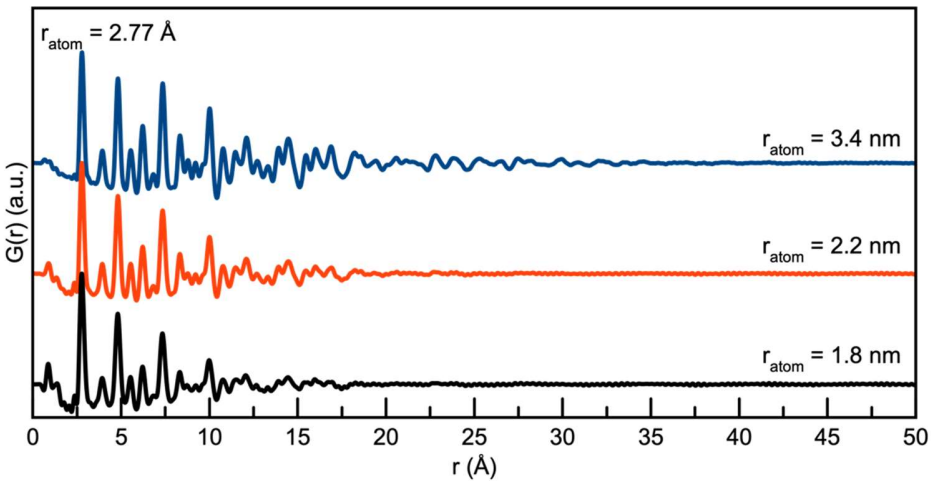

Fig. S2 | The 3 PDFs from Quinson et al..[14] The size estimate were determined by Quinson et al. by fitting the data using a spherical envelope.

The number of atoms in the particles were calculated as:

$$Atoms = \frac{V_{particle}}{V_{atom}} \cdot APF = \frac{{}^{4}/_{3} \cdot \pi \cdot {r_{particle}}^{3}}{{}^{4}/_{3} \cdot \pi \cdot {r_{atom}}^{3}} \cdot 0.740$$

Where V is the volume, r is the radius and APF is the atomic packing fraction which for *fcc* is 0.740. This yields a number of 203 atoms (1.8 nm), 371 atoms (2.2 nm) and 1368 atoms (3.4 nm).

G: MAE measures of reconstructed stacking faulted nanoparticles

Table S4 | Refined parameters for stacking faulted structure obtained from the DeepStruc algorithm for analysis of simulated PDFs. The MAEs for predicted xyz-coordinates is given along with the R_{wp}-values obtained in the PDF fit. The structures are shown in Fig. 6 of the main paper.

Name	**#Atoms**	**Scale**	**Expansion/ contraction factor**	**B_{iso}**	**MAE [Å]**	**MAE fit [Å]**	**R_{wp} [%]**
HCP	72	0.007	1.010	0.193	0.047 ± 0.026	0.038 ± 0.027	15.2
SF 1	72	0.007	1.003	0.266	0.026 ± 0.016	0.023 ± 0.015	8.6
SF 2	72	0.006	1.000	0.268	0.031 ± 0.014	0.031 ± 0.014	25.6
fcc	72	0.006	1.000	0.227	0.029 ± 0.011	0.029 ± 0.011	6.8

H: Simulation parameters of the PDFs

All PDFs used for conditioning DeepStruc were simulated using the DiffPy-CMI library.[20]

Table S5 | DiffPy-CMI simulation parameters for the PDFs of the seven structure types used in Fig. 1–5.

r_{min} (Å)	2.0	**Q_{min} (Å^{-1})**	0.7	**B_{iso} (Å^{-2})**	0.3

r_{max} (Å)	30.0	Q_{max} (Å^{-1})	25.0	**Delta2** (Å^{-2})	0.0
r_{step} (Å)	0.1	Q_{damp} (Å^{-1})	0.04		

Table S6 | DiffPy-CMI simulation parameters for the PDFs of the stacking faulted structures in Fig. 6.

r_{min} (Å)	0.0	Q_{min} (Å^{-1})	0.7	B_{iso} (Å^{-2})	0.0
r_{max} (Å)	20.0	Q_{max} (Å^{-1})	25.0	**Delta2** (Å^{-2})	0.0
r_{step} (Å)	0.1	Q_{damp} (Å^{-1})	0.04		

I: Normalisation of the PDFs

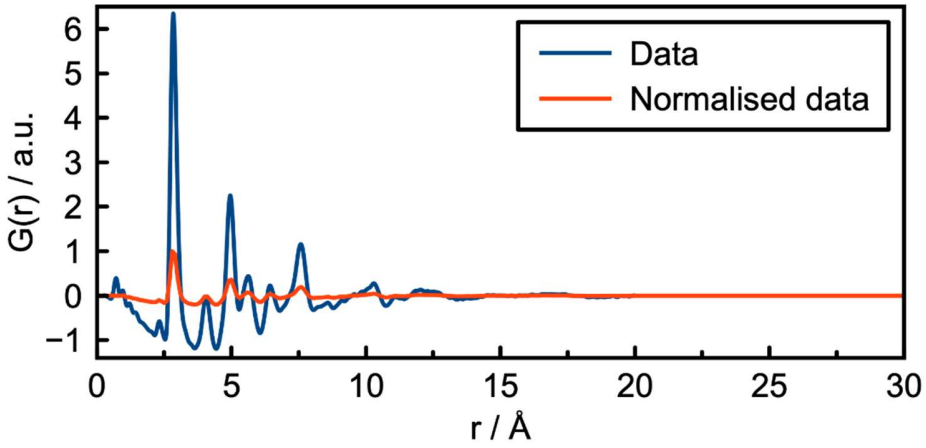

Fig. S6 | PDF and normalised PDF of the Au_{144}(*p*-MBA)$_{60}$ dataset from Jensen et al.[21]

J: Graph representation of MMNPs

Figure S7 shows a decahedron consisting of seven atoms alongside the components describing it in our chosen graph representation. Atoms 3 and 4 are connected through an edge, indicated by a yellow square in the adjacency matrix. Atoms being further separated than the lattice constant, such as atoms 3 and 5, do not have an edge

indicated by the black square in the adjacency matrix. Further, the edges in our graphs are undirected as we consider them to be bonds, hence the flow of information through bonded atoms are independent of direction.

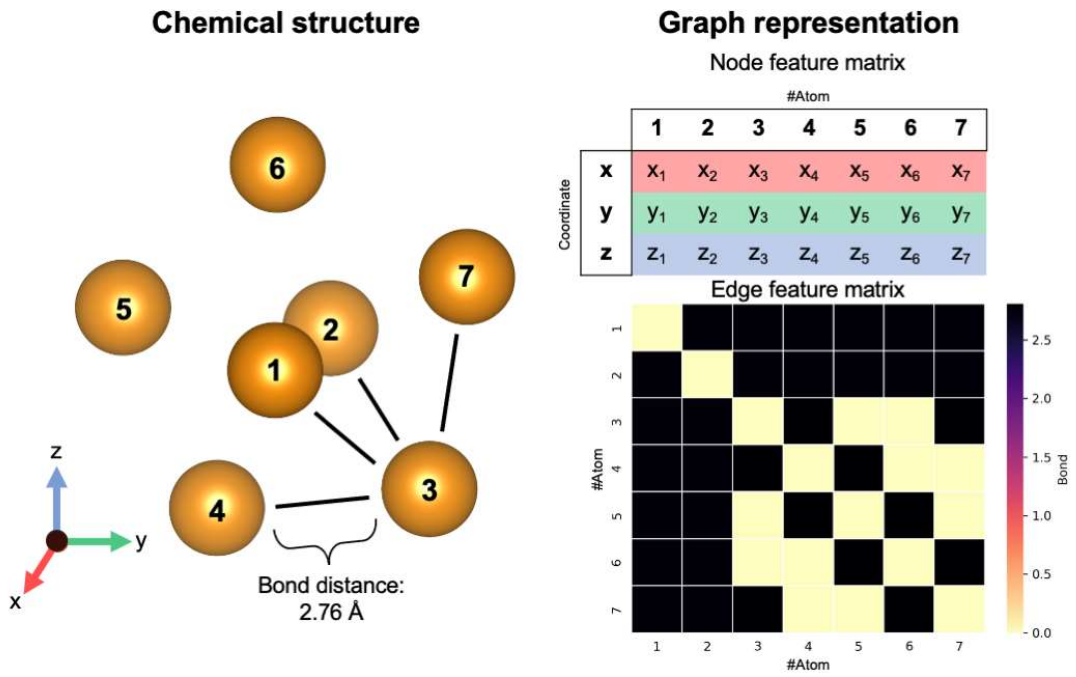

Fig. S7 | The chemical structure is shown in Euclidean space and has all atoms numbered. Atom number 3 has four neighbours each with a bond distance of approximately 2.76 Å. The graph representation is a mathematical approach to describe a chemical structure which maintains the interatomic relationship (edges) during translation, rotation and permutation. By adding the xyz-coordinates to the nodes and linking them through the adjacency matrix, the geometrical information can be transferred to the learning process of DeepStruc.

K: Structure determination from PDF: *fcc*, *hcp*, and stacking faulted nanoparticles

To obtain a deeper understanding of the latent space's behaviour, we investigate a dataset only containing *fcc*, *hcp,* and stacking faulted structures. *Fcc* and *hcp* structures are distinguished by the stacking sequence of closed packed layers in their structures: while *fcc* structures can be described by ABCABC stacking, *hcp* structures have ABABAB stacking. Structures with other sequences are stacking faulted structures. We

hypothesize that stacking faulted structures can be considered an 'interpolation' in the discrete space between the *fcc* and *hcp* structure type.[22]

Examples of reconstructed *fcc* (blue), *hcp* (pink), and different stacking faulted structures (purple) and their position in the new latent space are illustrated in Fig. 6a. The MMNPs cluster in size, whilst we also observe that *fcc* and *hcp* structures separate in the latent space. It is evident that the stacking faulted structures are located in between the *fcc* and *hcp* structures in the latent space as hypothesized. It is chemically reasonable that they are positioned in this exact order based on their similarity to *fcc* and *hcp*. For example, the structure with ABCABA layers, shown in Fig. 6 with a purple star is structurally close *fcc.* We see that it is also located closer to the *fcc* structures in the latent space. On the other hand, the structure with ABCBCB layers (marked as a purple diamond in Fig. 6) can be considered structurally more closely related to *hcp* than *fcc*. DeepStruc places this structure adjacent to *hcp* structures of the same size in the latent space. DeepStruc can thus insert stacking faulted structures between *fcc* and *hcp* into the latent space in a chemically meaningful way.

Fig. 6b illustrates the fits of the reconstructed structures to the PDF data. The difference curves indicate that the predicted and true structures are very close to being identical, which is supported by the MAE of the atomic positions on 0.030 ± 0.019 Å (section F in the Supplementary Information). While disorder causes a broadening of the peaks, the disorder in the generated structures is minor and structures with distinct difference between the layers and in the correct sequence can be reconstructed to a satisfying degree. This is a promising result, showing that a graph-based CVAE can be used as a tool to determine the structure of stacking faulted nanoparticles from PDFs,[23,24] which is a topic of significant current interest.[25-29]

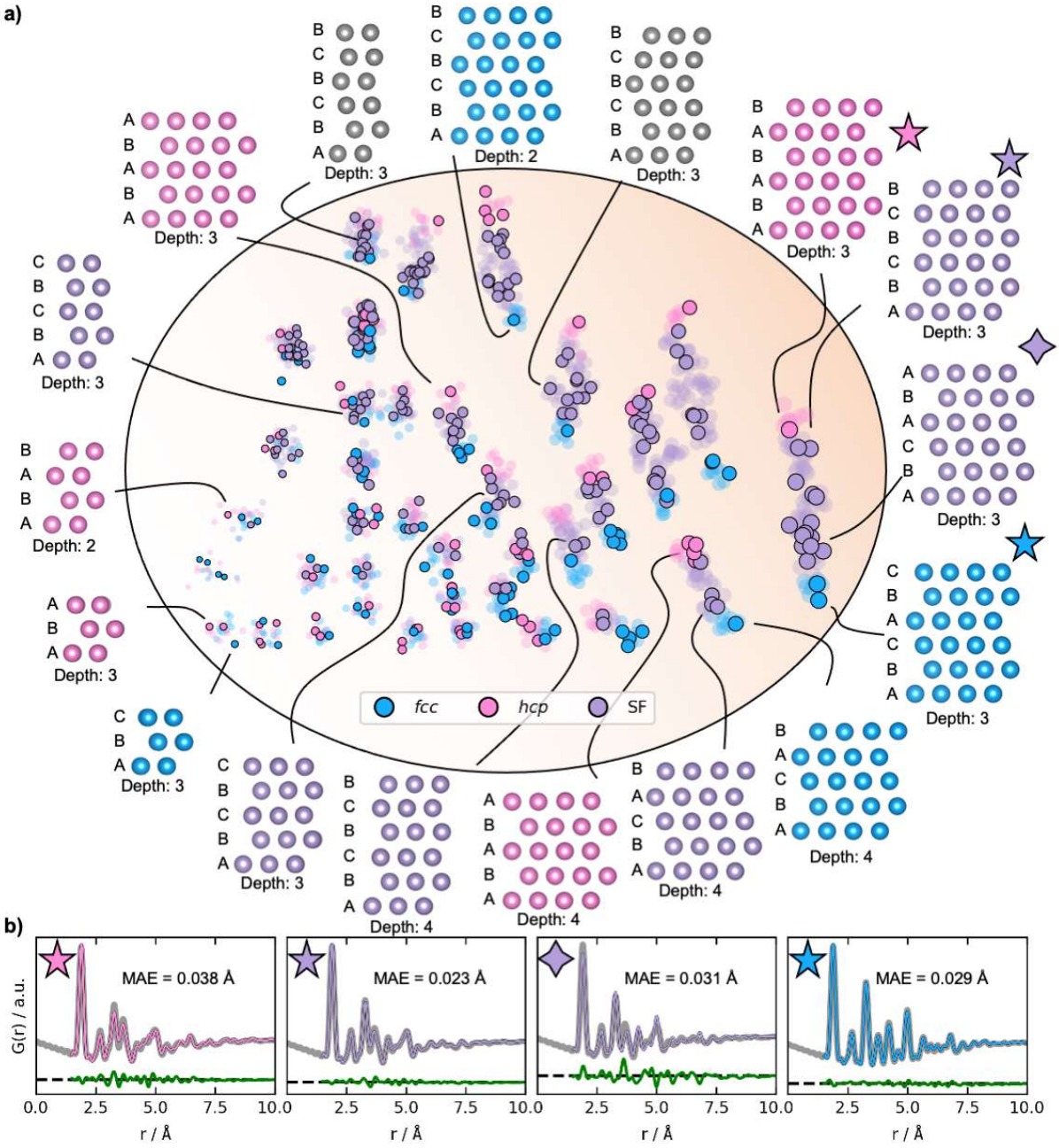

Fig. S8 | Latent space and reconstructions of stacking faulted nanoparticles. a) The latent space and reconstructed structures shown with their stacking sequence. The structures are shown in two dimensions, and the size (number of atoms) in the third dimension is given as 'depth'. The semi-transparent dots in the latent space represent the training and validation data, and the solid dots represent the test data. *Fcc* structures are

plotted in blue, *hcp* in pink, and the stacking faulted structures in purple. The marker size represents the size of the structures. B) Fits from reconstructed structures from the test PDF from a *fcc* (ABCABC stacking), a *hcp* (ABABAB stacking), and two stacking faulted structures. The original conditioning PDFs are shown in grey, while the PDFs of the generated structures are coloured according to their structure type. The difference curves are shown in green. The latent space is two-dimensional, hence allowing it to be directly visualized. Note that the test set structures shown here are the predicted structures obtained from DeepStruc during inference.

References

1 Egami, T. & Billinge, S. J. L. *Underneath the Bragg Peaks, Pergamon* (2012).

2 Hjorth Larsen, A. *et al.* The atomic simulation environment—a Python library for working with atoms. *J. Phys.: Condens. Matter* **29**, 273002, doi:10.1088/1361-648x/aa680e (2017).

3 Banerjee, S. *et al.* Cluster-mining: an approach for determining core structures of metallic nanoparticles from atomic pair distribution function data. *Acta Crystallogr. A* **76**, 24-31, doi:doi:10.1107/S2053273319013214 (2020).

4 Anker, A. S. *et al.* Characterising the Atomic Structure of Mono-Metallic Nanoparticles from X-Ray Scattering Data Using Conditional Generative Models. *Proceedings of the 16th International Workshop on Mining and Learning with Graphs (MLG)*, doi:10.26434/chemrxiv.12662222.v1 (2020).

5 Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A. & Vandergheynst, P. Geometric deep learning: going beyond euclidean data. *IEEE Signal Process. Mag.* **34**, 18-42 (2017).

6 Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M. & Monfardini, G. The graph neural network model. *IEEE Trans. Neural Netw.* **20**, 61-80 (2008).

7 Kingma, D. P. & Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

8 Sohn, K., Lee, H. & Yan, X. Learning structured output representation using deep conditional generative models. *Adv. Neural Inf. Process. Syst.* **28**, 3483-3491 (2015).

9 Lim, J., Ryu, S., Kim, J. W. & Kim, W. Y. Molecular generative model based on conditional variational autoencoder for de novo molecular design. *J. Cheminformatics* **10**, 1-9 (2018).

10 Duxbury, P. M., Granlund, L., Gujarathi, S., Juhas, P. & Billinge, S. J. The unassigned distance geometry problem. *Discret. Appl. Math.* **204**, 117-132 (2016).

11 Shao, H. *et al.* ControlVAE: Tuning, Analytical Properties, and Performance Analysis. *IEEE Trans. Pattern Anal. Mach. Intell.* (2021).

12 Rydhmer, K. & Selvan, R. Dynamic $\beta $-VAEs for quantifying biodiversity by clustering optically recorded insect signals. *arXiv preprint arXiv:2102.05526* (2021).

13 Kipf, T. N. & Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

14 Quinson, J. *et al.* Spatially Localized Synthesis and Structural Characterization of Platinum Nanocrystals Obtained Using UV Light. *ACS Omega* **3**, 10351-10356, doi:10.1021/acsomega.8b01613 (2018).

15 Tomczak, J. M. & Welling, M. Improving variational auto-encoders using householder flow. *arXiv preprint arXiv:1611.09630* (2016).

16 Jensen, K. M. Ø. *et al.* Polymorphism in magic-sized Au144(SR)60 clusters. *Nat. Commun.* **7**, doi:10.1038/ncomms11859 (2016).

17 Juhas, P., Farrow, C. L., Yang, X., Knox, K. R. & Billinge, S. J. L. Complex modeling: a strategy and software program for combining multiple information sources to solve ill posed structure and nanostructure inverse problems. *Acta Crystallogr. A* **71**, 562-568, doi:doi:10.1107/S2053273315014473 (2015).

18 Chen, T. & Guestrin, C. in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 785–794 (Association for Computing Machinery, San Francisco, California, USA, 2016).

19 Anthony, L. F. W., Kanding, B. & Selvan, R. Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. *arXiv preprint arXiv:2007.03051* (2020).

20 Juhas, P., Farrow, C. L., Yang, X., Knox, K. R. & Billinge, S. J. L. Complex modeling: a strategy and software program for combining multiple information sources to solve ill posed structure and nanostructure inverse problems. *Acta Cryst.* **71**, 562-568, doi:10.1107/S2053273315014473 (2015).

21 Jensen, K. M. Ø. *et al.* Polymorphism in magic-sized $Au_{144}(SR)_{60}$ clusters. *Nat Commun* **7**, 11859, doi:10.1038/ncomms11859 (2016).
22 Bertolotti, F. *et al.* A total scattering Debye function analysis study of faulted Pt nanocrystals embedded in a porous matrix. *Acta Crystallogr. A* **72**, 632-644, doi:doi:10.1107/S205327331601487X (2016).
23 Masadeh, A. S. *et al.* Quantitative size-dependent structure and strain determination of CdSe nanoparticles using atomic pair distribution function analysis. *Phys. Rev. B* **76**, doi:Artn 115413
Doi 10.1103/Physrevb.76.115413 (2007).
24 Yang, X. *et al.* Confirmation of disordered structure of ultrasmall CdSe nanoparticles from X-ray atomic pair distribution function analysis. *Phys. Chem. Chem. Phys.* **15**, 8480-8486, doi:10.1039/C3CP00111C (2013).
25 Cenker, J. *et al.* Reversible strain-induced magnetic phase transition in a van der Waals magnet. *Nat. Nanotechnol.*, doi:10.1038/s41565-021-01052-6 (2022).
26 Rong, X. *et al.* Structure-Induced Reversible Anionic Redox Activity in Na Layered Oxide Cathode. *Joule* **2**, 125-140, doi:https://doi.org/10.1016/j.joule.2017.10.008 (2018).
27 Charles, D. S. *et al.* Structural water engaged disordered vanadium oxide nanosheets for high capacity aqueous potassium-ion storage. *Nat. Commun.* **8**, 15520, doi:10.1038/ncomms15520 (2017).
28 Gao, P. *et al.* The critical role of point defects in improving the specific capacitance of δ-MnO2 nanosheets. *Nat. Commun.* **8**, 14559, doi:10.1038/ncomms14559 (2017).
29 Metz, P. C., Koch, R. & Misture, S. T. Differential evolution and Markov chain Monte Carlo analyses of layer disorder in nanosheet ensembles using total scattering. *J. Appl. Crystallogr.* **51**, 1437-1444, doi:doi:10.1107/S1600576718011597 (2018).