Supplementary Information to Neural networks trained on synthetically generated crystals can extract structural information from ICSD powder X-ray diffractograms

Henrik Schopmans, Patrick Reiser, and Pascal Friederich*

S1 Generating synthetic crystals

Here we describe the algorithm to generate synthetic crystals in more detail. To generate a single crystal, the following steps are executed:

1. Sampling of a space group from the space group distribution of the ICSD.
2. The number of unique elements in the crystal is drawn from a discrete distribution extracted from the crystals in the ICSD belonging to the specified space group.
3. The unique elements are drawn, also from a discrete probability distribution from the crystals in the ICSD belonging to the specified space group.
4. For each of the unique elements, the number of repetitions in the asymmetric unit is chosen, and for each repetition, a Wyckoff position is selected. Again, both the probability of the number of repetitions and the Wyckoff occupation probabilities are extracted from the ICSD for the specified space group. We do not place more than one atom onto a Wyckoff position that does not have a degree of freedom.
5. For each atom placed on a Wyckoff position, uniformly distributed random fractional coordinates are drawn.
6. Lattice parameters (normalized to unity volume) of the crystal system that the specified space group belongs to are drawn from a kernel density estimate of the ICSD. The bandwidth is chosen based on Scott's rule (see the *SciPy*[1] implementation of the kernel density estimate).
7. We generated a kernel density estimate of the volume conditioned[1] on $\sum_i 4/3\pi \left(\frac{r_{i;\mathrm{cov}}+r_{i;\mathrm{VdW}}}{2}\right)^3 = V_{\mathrm{atomic}}$, where the sum covers all atoms in the conventional unit cell, $r_{i;\mathrm{cov}}$ is the atomic covalent radius, and $r_{i;\mathrm{VdW}}$ is the atomic van der Waals radius. The kernel density estimate was generated from all crystals of the ICSD belonging to the specified space group. Then, V_{atomic} is calculated for the chosen atoms in the conventional unit cell and the volume is drawn based on the kernel density estimate conditioned on V_{atomic}. The lattice parameters (chosen in the previous step) are further scaled by the cube root of the chosen volume.
8. Space group symmetry operations are applied using *Python* library *PyXtal*[3].

When generating a crystal of a specific space group without placing an atom on the general Wyckoff position, it is not always the case that the crystal belongs to that space group. To prevent wrong space group assignments, we use the *Pymatgen*[4] interface to *Spglib*[5] to check the space group of each crystal after its generation. If the space group deviates, we generate a new crystal with the same number of unique elements as before, in order to not distort the distribution of number of unique elements extracted from the ICSD. If the generation fails 20 times in total, we start from the beginning with a new number of unique elements.

[1]For this conditional kernel density estimate, we used the implementation of *statsmodels*[2] with the normal reference rule of thumb to estimate the bandwidth.

S2 Machine learning

S2.1 Models

We now want to describe the machine learning models that we used for the classification of space groups in more detail. Powder diffractograms include similar features (peaks) at different locations and the position of a feature in the diffractogram has a spatial meaning. This suggests that the properties of the convolution operation, namely the parameter sharing (with sparse connectivity) and equivariance[6], might be beneficial when processing powder diffractograms.

As a baseline, we first used the two CNN architectures used by Park *et al.*[7] for the classification of extinction groups and space groups. Since our training dataset is an infinite stream of diffractograms and we do not have to worry about overfitting, we further used the deeper architectures ResNet-10, ResNet-50, and ResNet-101[8]. All architectures are now described in detail.

Architectures by Park *et al.*

Park *et al.*[7] introduced three models, one for the classification of the crystal system, one for extinction groups, and one for space groups. We used only the last two models and call them "parkCNN medium" and "parkCNN big", respectively.

"parkCNN big" consists of three convolution layers with average pooling, two hidden fully connected layers with 2300 and 1150 nodes, and a 145-dimensional softmax output. The architecture is summarized in Figure S1. The "parkCNN medium" model has fewer parameters than "parkCNN big" since the two hidden fully connected layers have 4040 and 202 nodes.

ResNet architecture

With increasing size of the training dataset and increasing difficulty of the chosen task, the number of model parameters needs to be increased, too.

In principle, a deeper model with additional layers should always be able to express the same solution of a shallower model by simply "learning" an identity map in addition to the shallower model. In practice, however, a degradation problem for CNNs with increasing depth has been observed and very deep models can perform worse than their shallow counterpart[8]. Therefore, the ResNet architecture developed by He *et al.*[8] at Microsoft in 2015 introduced additional skip connections, where information is able to simply flow past the convolution layer and is added to its output. This makes it possible for needed information of the input or earlier layers to flow further into the model without degradation.

Figure S2a visualizes the residual block used for the shallower versions of the ResNet architecture (up to 34 layers). Figure S2b visualizes the bottleneck block used for the deeper variants (50 and more layers). This type of building block is called a bottleneck block since it first reduces the number of channels using a 1x1 convolution operation with N filters. Then, the main convolution with a 3x3 kernel is performed, followed by a third 1x1 convolution that upscales to 4N channels. This down- and upscaling of the number of channels is performed to increase the performance of the model. All convolution operations of both types of building blocks are followed by batch normalization implicitly.

In the simplest case, the skip connection of the residual and bottleneck block is simply an identity mapping and added to the output of the block. However, if the number of input channels and dimensions of a block are different from those in the output, a projection in the form of a 1x1 convolution with the necessary number of filters and stride (usually 2) is used instead of the identity.

Table S1 summarizes the ResNet-10, ResNet-50, and ResNet-101 models. Each architecture is to be read from top to bottom. The square brackets indicate a residual or bottleneck block with their two or three convolution operations and respective number of filters. The number after the square brackets "$\times N$" indicates how often the building block is to be repeated in the respective block group.

If the output dimension changes from one block group to the next, the first building block of the next block group downsamples the dimensions by using a stride of 2 for the first 3x3 convolution in the case of a residual block and for the middle 3x3 convolution in the case of a bottleneck block. All other convolution operations of the building blocks are performed with stride 1.

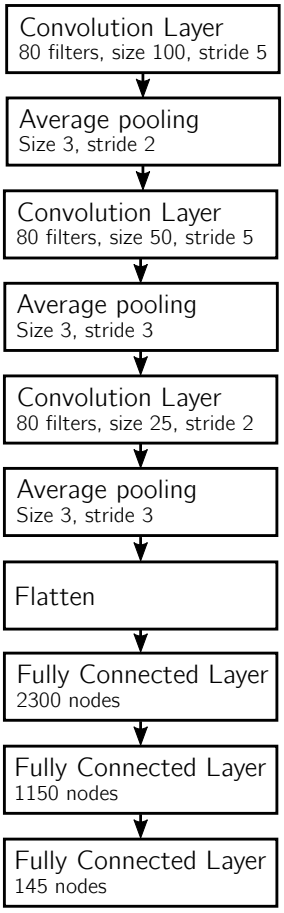

Figure S1: The CNN architecture for space group classification ("parkCNN big") as introduced by Park *et al.*[7]. Each convolution or fully connected layer is implicitly followed by a ReLU activation, the output uses a softmax activation. We used only 145 target space groups instead of 230, since the remaining space groups did not have enough entries present in the ICSD to extract enough statistics for the synthetic generation of crystals. Furthermore, Park *et al.* used an input length of 10001 instead of our input length of 8501. A dropout rate of 30% is used after the activations of the convolution blocks. Dropout with a rate of 50% is used after the activations of the fully connected layers. However, dropout is only used if the model is directly trained on ICSD diffractograms, not when using the synthetic data.

We used the ResNet implementation as found in the *TensorFlow* model garden[9]. Since our data is one-dimensional, we used an adapted 1D version. We replaced all 2D convolutions and pooling operations with their 1D equivalent ($N \times N \to N$). Furthermore, we used a kernel size of 9 in place of the 3x3 kernels and stride 4 instead of stride 2 in the bottleneck blocks and projection skip connections ($N \times N \to N^2$). This squaring was performed to obtain a better distribution of the number of weights throughout the architecture (similar to the original 2D case). We further added an additional fully connected layer with 256 nodes after the flatten layer in the end of the ResNet models, followed by the output layer.

We were not able to achieve good results when using the original ResNet architecture with batch normalization (similar observations were made by Schuetzke *et al.*[10]). The test accuracy calculated after each epoch was highly unstable and had high fluctuations, probably in part due to the moving averages of the batch normalization not converging properly. This is possibly caused by using an infinite stream of batches of diffractograms, instead of using a training dataset of fixed size. We fixed this problem by using group normalization[11] with 32 groups instead of batch

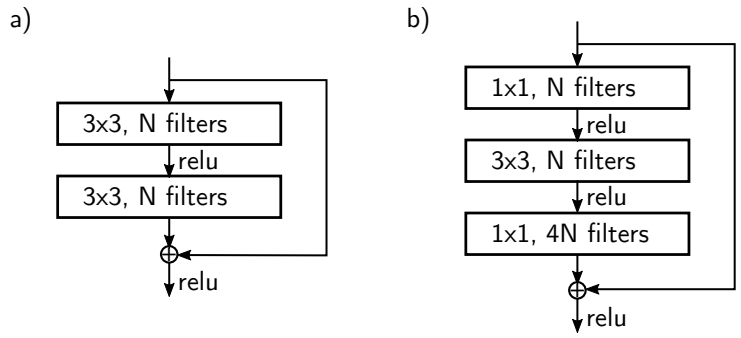

Figure S2: a) ResNet residual block, the main building block of the shallower variants of the ResNet architecture. b) ResNet bottleneck block, the main building block of the deeper variants of the ResNet architecture. All convolution operations are implicitly followed by a batch normalization layer. In both cases, a skip connection allows information to directly flow past the convolution operations. (Illustration based on[8])

Table S1: Composition of the ResNet-10, ResNet-50, and ResNet-101 architectures[8]. The architectures are to be read from top to bottom. Square brackets indicate a residual or bottleneck building block with the respective number of filters for each convolution.

output size	layer name	ResNet-10	ResNet-50	ResNet-101
112x112	initial conv.	7×7 conv., 64 channels, stride 2		
56x56	initial pool	3×3 max pool, stride 2		
56x56	block group 1	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times1$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times3$
28x28	block group 2	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times1$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times4$
14x14	block group 3	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times1$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times23$
7x7	block group 4	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times1$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times3$

normalization.

S2.2 Setup

We used a distributed architecture on multiple nodes using the *Python* framework *Ray*[12]. The configuration utilized throughout this study is visualized in Figure 2b of our paper. Training took place on a *Ray* head node with one or two[2] RTX 2080 Ti GPUs. Training on two GPUs was performed using the *MirroredStrategy* of *TensorFlow*[13]. 28 out of the 32 cores of the head node were used for the generation of diffractograms. Next to the head node, we used two additional compute nodes with 128 cores each to generate diffractograms. Communication between the nodes took place using a *Ray queue* object to access the simulated diffractograms.

Depending on the model size and corresponding training speed, this setup allows training with up to millions of unique diffractograms per hour. For the larger ResNet variants ResNet-50 and ResNet-101, we efficiently used the GPUs. For the "parkCNN" and ResNet-10 models, training is

[2]The "parkCNN" models used a single GPU, while the ResNet models were trained on two GPUs.

fast and the data generation becomes the bottleneck.

To train all models, we used *Keras*[14] with *TensorFlow* 2.3[13]. Optimization was performed using *Adam*[15] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ (*Keras* default parameters). We also tried to use stochastic gradient descent (SGD) with momentum and a stepwise learning rate scheduler, but this did not yield good results. Depending on the initial conditions, most of the training runs using SGD were either unstable or reached low accuracies. For all experiments, a cross-entropy loss was utilized.

We used a batch size of 870 for the "parkCNN" models and a batch size of 145 for the three ResNet models. Furthermore, the "parkCNN" models were trained for 1000 epochs with a learning rate of 0.001, while the ResNet models were trained for 2000 epochs with a learning rate of 0.0001. The ResNet models used a step decay of the initial learning rate, halving the learning rate after every 500 epochs.

When training on diffractograms simulated from synthetic crystals, we used 150 generated batches per epoch when a batch size of 870 was used ("parkCNN medium" and "parkCNN big") and 900 batches per epoch when a batch size of 145 was used (ResNet). This means that each epoch always contained 130 500 diffractograms[3].

For the experiments performed directly on diffractograms simulated from ICSD crystals, we used the statistics dataset directly to pre-generate the training dataset. We excluded the same 85 space groups that were not used by the synthetic training due to missing statistics. We used two different crystallite sizes for each crystal in the statistics dataset, yielding $148\,466 \cdot 2 = 296\,932$ diffractograms in the training dataset.

S3 Randomized datasets

In Section 3.3 of the main text, we created randomized variations of the test and statistics datasets to analyze and understand the gap between training on synthetic crystals and testing on the ICSD. For each of the two datasets, we used 22 500 randomly picked crystals for the analysis. This analysis used the space group labels as reported by the library *PyXtal* (and not the labels as reported by the ICSD, which can deviate for a small number of structures). Furthermore, *PyXtal* does not support partial occupancies. Therefore, we compared the accuracies we obtained on the randomized datasets with a dataset that has all occupancies set to 1.0 and uses the *PyXtal* space group labels. We call this the "reference dataset". The difference in accuracy between this reference dataset and the test / statistics dataset is relatively small (≈ 1 percentage point).

S4 Application to experimental diffractograms

S4.1 Dataset

To test the performance on experimental diffractograms, we used the publicly available RRUFF mineral database[16]. It contains 5829 mineral samples with multiple types of measured spectra and, most important for us, powder XRD measurements with K_α radiation for 2952 samples. Of these 2952 samples, 2875 samples provide the output of a Rietveld refinement, including the space group label. We further removed by hand some samples that had excessive amounts of noise and were of bad quality. Many samples further only provide a simulated diffractogram without noise or background. They were also excluded. This left 942 diffractograms for our analysis.

S4.2 Data generation

To be able to apply our models to experimental diffractograms, we added an additional background function and noise to the generated diffractograms to make them more similar to experimental data. We used a Gaussian process to generate random background functions and added additive and multiplicative Gaussian noise. All diffractograms were generated in the range $2\theta \in [5, 90]^\circ$ with step size 0.01°.

The generation protocol is as follows:

[3]Note that the division into epochs for training using synthetic crystals is rather arbitrary since each epoch contains different diffractograms simulated from different crystals. However, the division makes it easy to calculate the performance on the test dataset after each epoch.

1. Sample the background function from a Gaussian process with radial basis function kernel without any conditioning:

$$k(x_i, x_j) = a \exp\left(-\frac{|x_i - x_j|^2}{2l^2}\right) \tag{1}$$

   We chose $a = 1$ and sampled l uniformly in $[7, 40]$ for each diffractogram.

2. Subtract the minimum intensity from the background function obtained from the Gaussian process.

3. Divide it by the sum of the 8501 (2θ-range) entries.

4. Multiply it by a scaling factor drawn from a truncated normal distribution in the range $[0, 150]$ with mean 0 and standard deviation 75.

5. Add the pure diffractogram (intensity-range $[0, 1.0]$).

6. Add Gaussian additive noise with mean 0 and standard deviation uniformly drawn in $[0, 0.02]$.

7. Multiply with multiplicative Gaussian noise with mean 1 and standard deviation uniformly drawn in $[0, 0.16]$.

To resemble a more realistic peak profile, we used the pseudo-Voigt profile instead of the Gaussian profile that we used for the classification of pure diffractograms. The pseudo-Voigt profile uses the full width at half maximum (FWHM) Γ_G of the Gaussian G, the FWHM Γ_L of the Lorentzian L, and a mixing parameter η as parameters[17]:

$$\mathrm{PV}\left(\theta - \theta_0; \Gamma_\mathrm{L}, \Gamma_\mathrm{G}\right) = \eta G\left(\theta - \theta_0; \Gamma_\mathrm{G}\right) + (1 - \eta) L\left(\theta - \theta_0; \Gamma_\mathrm{L}\right) \tag{2}$$

with

$$G(\theta - \theta_0; \sigma = \frac{\Gamma_\mathrm{G}}{2\sqrt{2\ln 2}}) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\theta - \theta_0}{\sigma}\right)^2} \tag{3}$$

and

$$L(\theta - \theta_0; \gamma = \frac{1}{2}\Gamma_\mathrm{L}) = \frac{1}{\pi\gamma}\left[\frac{\gamma^2}{\left(\theta - \theta_0\right)^2 + \gamma^2}\right] . \tag{4}$$

The FWHM Γ_G of the Gaussian is typically parametrized using the Caglioti equation[17] as

$$\Gamma_\mathrm{G}^2 = U \tan^2\theta + V \tan\theta + W . \tag{5}$$

Following the suggestions for typical Rietveld parameter ranges by Kaduk & Reid[18] and comparing the resulting peaks with the ones from the RRUFF dataset, we decided to sample the Caglioti parameters uniformly in the following ranges: $[0, 0.01]$ for U, $[0, 0.01]$ for W, and V was fixed at $V = 0.0$. We further used a single mixing parameter η uniformly sampled in $[0.0, 1.0]$ for the full 2θ-range. For simplicity, we used the same FWHM from the Caglioti equation for both the Gaussian and Lorentzian of the pseudo-Voigt. We further considered the K_{α_1} / K_{α_2} splitting of the K_α line since for some diffractograms in the RRUFF dataset, this splitting is visible.

We further implemented the option to add impurity phases to the training (and simulated ICSD test) data. The minerals of the RRUFF database are not all made up of one phase, but most of them contain small amounts of one or more impurity phases. To model this, for each training diffractogram, we used a superposition of the main phase to be classified and an impurity phase of a random space group (a is uniformly sampled in $[0, 0.05]$):

$$I(\theta) = (1 - a) I_\mathrm{pure} + a I_\mathrm{impurity} \tag{6}$$

S4.3 Experiments

For our experiments on experimental data, we used the same split based on structure types as we used for pure diffractograms. We performed two experiments using the ResNet-50 architecture, one with impurity phases and one without. For both, a learning rate of 0.0001, a batch size of 145, and 1000 epochs where used.

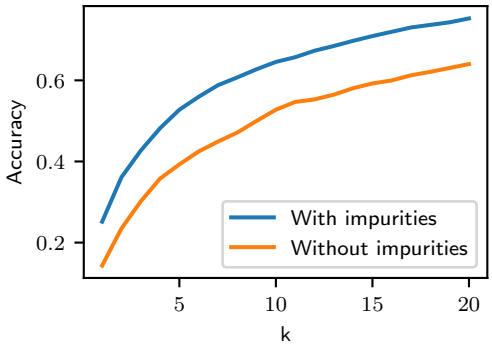

Figure S3: Top-k accuracy as a function of k tested on RRUFF dataset for ResNet-50 model trained with added noise and background. The experiment corresponding to the blue curve additionally contained one added impurity phase for each diffractogram in the training data.

S5 Additional figures and tables

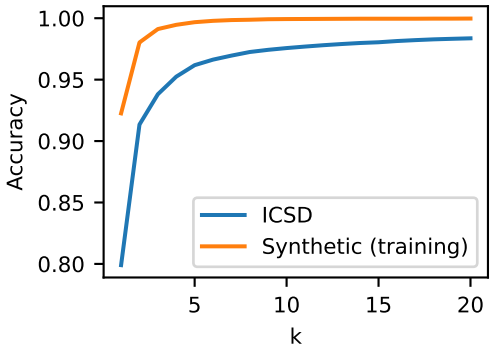

Figure S4: Top-k accuracy as a function of k tested on ICSD test dataset and the synthetic training data for ResNet-101 model trained on synthetic data.

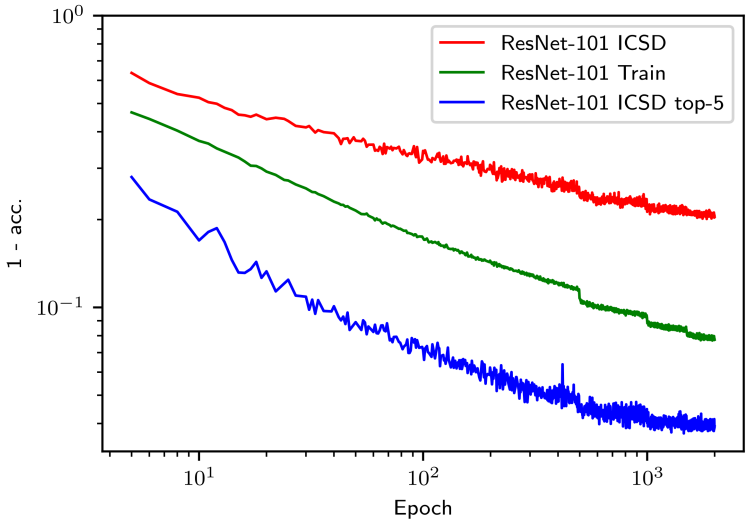

Figure S5: $1 - \text{acc.}$ for test accuracy (ICSD), training accuracy (synthetic crystals), and test top-5 accuracy (ICSD) as a function of epochs for ResNet-101 model trained on synthetic data. To better show the scaling behavior, both axes use logarithmic scaling.

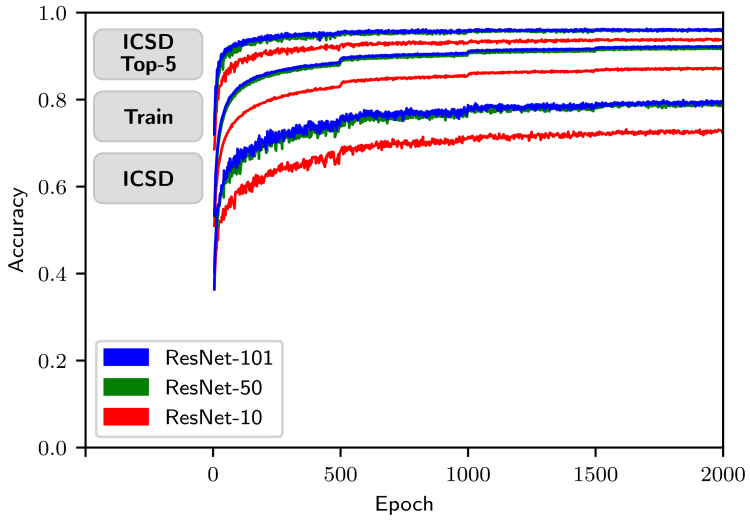

Figure S6: Test accuracy (ICSD), training accuracy (synthetic crystals), and test top-5 accuracy (ICSD) as a function of epochs. We show all three metrics for the models ResNet-101, ResNet-50, and ResNet-10.

Table S2: This is an extension of Table 1 of the main text. We additionally provide the total number of unique diffractograms seen during training and the training time for each computational experiment. To obtain the total number of unique diffractograms, we also counted diffractograms that are based on the same crystal structure but have a different crystallite size. To get the number of unique crystals, the provided number for all experiments directly trained on ICSD data and for the experiment using synthetic data with the "parkCNN big" model needs to be divided by two, since each crystal is used to generate two diffractograms with different crystallite sizes in those experiments (see Section 2.5 of the main text). Training times are based on the hardware setup described in Section S2.2.

Split	Training dataset	Testing dataset	Model	Total number of unique diffractograms	Training time
Random	ICSD	ICSD	parkCNN medium	296 932	$\approx$ 1 day
Structure type	ICSD	ICSD	parkCNN big	296 932	$\approx$ 1 day
			parkCNN medium	296 932	$\approx$ 1 day
Structure type[1]	synthetic	ICSD	parkCNN big	130 500 000	$\approx$ 1 day
			ResNet-10	261 000 000	$\approx$ 3.5 days
			ResNet-50	261 000 000	$\approx$ 11 days
			ResNet-101	261 000 000	$\approx$ 17.5 days

[1]Here, the split type refers to the statistics and the test dataset, rather than the training and the test dataset.

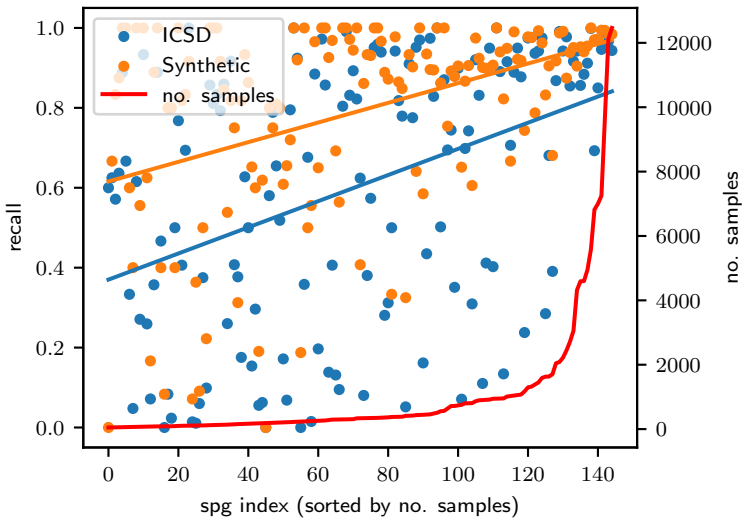

Figure S7: Using the ResNet-101 model trained on the synthetic crystals, this figure shows the recall ($\frac{TP}{TP+FN}$) and no. samples for all 145 space groups included in our experiments. Space group numbers are sorted by the no. samples. In blue, one can find the recall tested on the ICSD test dataset, in orange tested on diffractograms from the synthetic training distribution. The blue and orange lines show the trend of the recall.

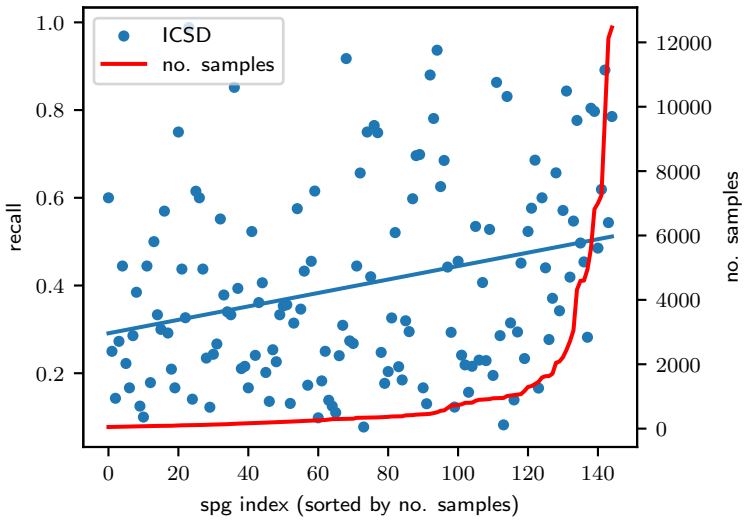

Figure S8: Using the "parkCNN big" model trained on the ICSD crystals directly, this figure shows the recall ($\frac{TP}{TP+FN}$) and no. samples for all 145 space groups included in our experiments. Space group numbers are sorted by the no. samples. The recall is tested on the ICSD test dataset. The blue line shows the trend of the recall.

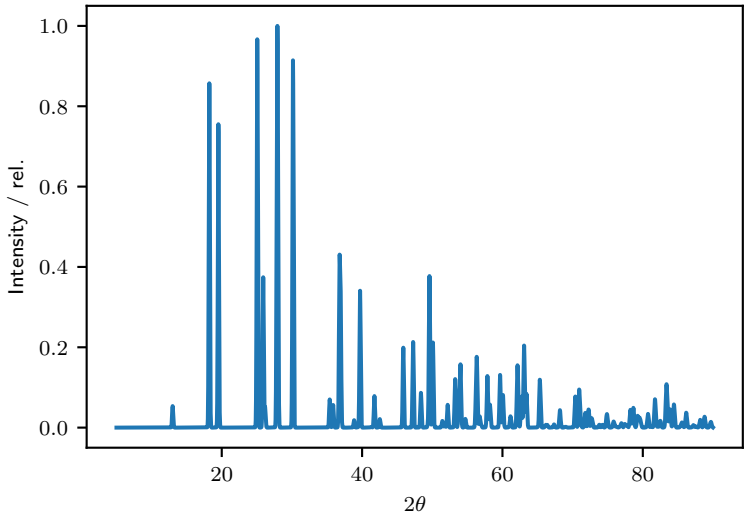

Figure S9: Exemplary diffractogram from the ICSD, simulated without imperfections (noise and background).

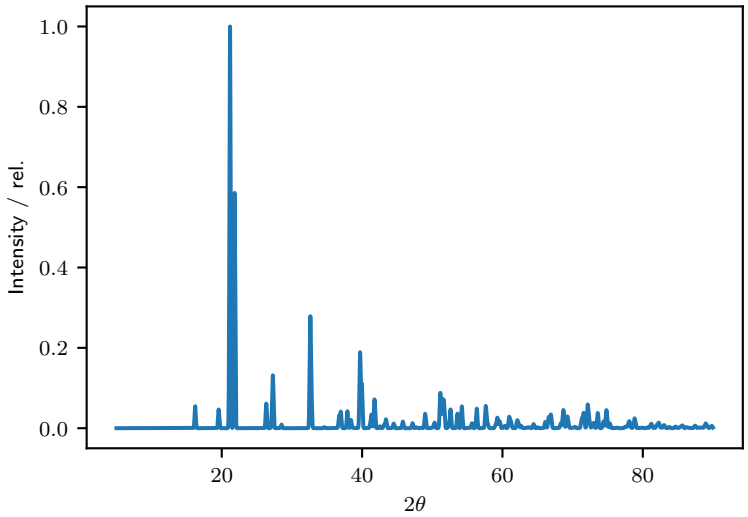

Figure S10: Exemplary diffractogram from the synthetic crystal distribution, simulated without imperfections (noise and background).

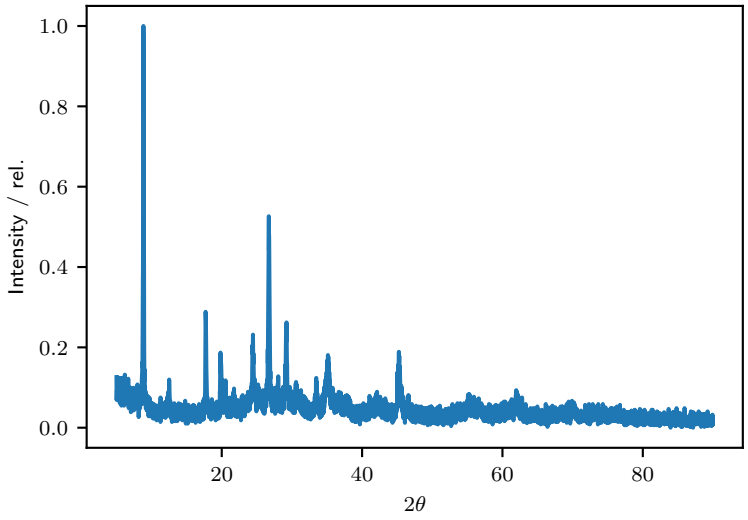

Figure S11: Exemplary diffractogram from experimental RRUFF mineral database.

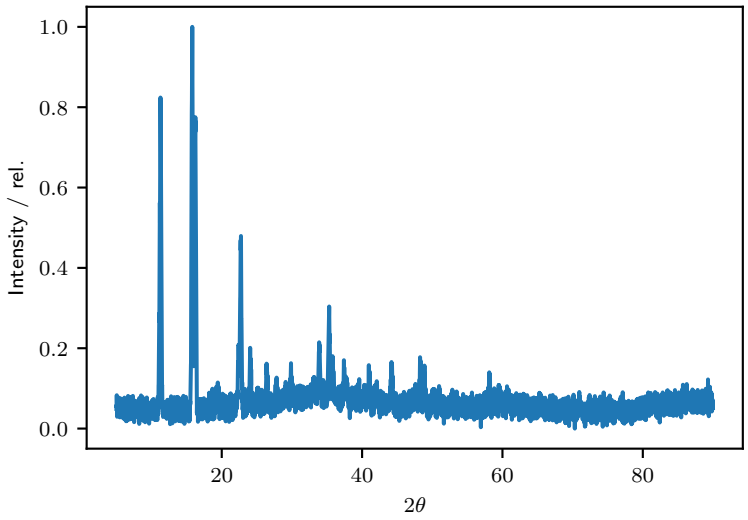

Figure S12: Exemplary diffractogram from the synthetic crystal distribution, simulated with imperfections (noise, background, and impurity phase).

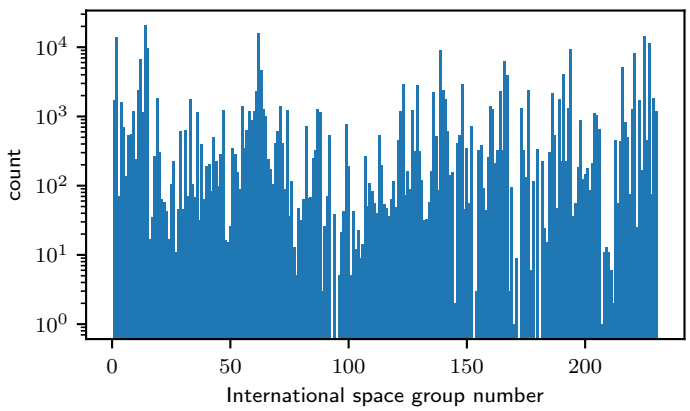

Figure S13: Distribution (logarithmic scale) of space groups in the ICSD.

References

1. Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F. & van Mulbregt, P. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nat Methods* **17,** 261–272. ISSN: 1548-7105. doi:10.1038/s41592-019-0686-2. (2022-08-31) (3 2020-03).
2. Seabold, S. & Perktold, J. Statsmodels: Econometric and Statistical Modeling with Python. *Proceedings of the 9th Python in Science Conference* **2010** (2010-01-01).
3. Fredericks, S., Parrish, K., Sayre, D. & Zhu, Q. PyXtal: A Python Library for Crystal Structure Generation and Symmetry Analysis. *Computer Physics Communications* **261,** 107810. ISSN: 0010-4655. doi:10.1016/j.cpc.2020.107810. (2022-06-27) (2021-04-01).
4. Ong, S. P., Richards, W. D., Jain, A., Hautier, G., Kocher, M., Cholia, S., Gunter, D., Chevrier, V. L., Persson, K. A. & Ceder, G. Python Materials Genomics (Pymatgen): A Robust, Open-Source Python Library for Materials Analysis. *Computational Materials Science* **68,** 314–319. ISSN: 0927-0256. doi:10.1016/j.commatsci.2012.10.028. (2022-06-21) (2013-02-01).
5. Togo, A. & Tanaka, I. *Spglib: A Software Library for Crystal Symmetry Search* arXiv: 1808.01590. (2022-06-28). preprint.
6. Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning* 800 pp. http://www.deeplearningbook.org (The MIT Press, 2016).
7. Park, W. B., Chung, J., Jung, J., Sohn, K., Singh, S. P., Pyo, M., Shin, N. & Sohn, K.-S. Classification of Crystal Structure Using a Convolutional Neural Network. *IUCrJ* **4,** 486–494. ISSN: 2052-2525. doi:10.1107/S205225251700714X. (2021-09-06) (4 2017-07-01).
8. He, K., Zhang, X., Ren, S. & Sun, J. *Deep Residual Learning for Image Recognition* in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016-06), 770–778. doi:10.1109/CVPR.2016.90.
9. Yu, H., Chen, C., Xianzhi, D., Yeqing, L., Abdullah, R., Le, H., Pengchong, J., Fan, Y., Frederick, L., Jaeyoun, K. & Jing, L. *TensorFlow Model Garden* doi:10.5281/zenodo.11813. (2022-08-18).
10. Schuetzke, J., Szymanski, N. J. & Reischl, M. *A Universal Synthetic Dataset for Machine Learning on Spectroscopic Data* arXiv: 2206.06031 [cond-mat]. (2023-03-24). preprint.
11. Wu, Y. & He, K. *Group Normalization* in. Proceedings of the European Conference on Computer Vision (ECCV) (2018), 3–19. doi:10.1007/s11263-019-01198-w.
12. Moritz, P., Nishihara, R., Wang, S., Tumanov, A., Liaw, R., Liang, E., Elibol, M., Yang, Z., Paul, W., Jordan, M. I. & Stoica, I. *Ray: A Distributed Framework for Emerging AI Applications* in. 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18) (2018), 561–577. ISBN: 978-1-939133-08-3. https://www.usenix.org/conference/osdi18/presentation/moritz (2022-09-13).
13. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. & Zheng, X. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems* 2015. https://www.tensorflow.org/.
14. François, C. *Keras* https://keras.io.
15. Kingma, D. P. & Ba, J. *Adam: A Method for Stochastic Optimization* arXiv: 1412.6980. (2022-06-28). preprint.

16. Lafuente, B., Downs, R. T., Yang, H. & Stone, N. *The Power of Databases: The RRUFF Project in Highlights in Mineralogical Crystallography* ISBN: 978-3-11-041710-4 (De Gruyter (O), 2015-11-13).

17. *International Tables for Crystallography Volume H: Powder Diffraction* 1st ed. (eds Gilmore, C. J., Kaduk, J. A. & Schenk, H.) ISBN: 978-1-118-41628-0 (Wiley, 2019-09-24).

18. Kaduk, J. & Reid, J. Typical Values of Rietveld Instrument Profile Coefficients. *Powder Diffraction - POWDER DIFFR* **26,** 88–94. doi:10.1154/1.3548128 (2011-03-01).