Deep learning for x-ray or neutron scattering under grazing-incidence: extraction of distributions

Walter Van Herck1,3, Jonathan Fisher2 and Marina Ganeva1,*

1 Forschungszentrum Jülich GmbH, Jülich Centre for Neutron Science (JCNS) at Heinz Maier-Leibnitz Zentrum (MLZ), Lichtenbergstrasse 1, 85748 Garching, Germany
2 Cape Analytics GmbH, München, Germany
3 Presently at ITER Organization, Route de Vinon-sur-Verdon, CS 90 046, 13067 St. Paul Lez Durance Cedex, France.

* Author to whom any correspondence should be addressed.
E-mail: m.ganeva@fz-juelich.de

Keywords: GISAS, nanoparticles, deep learning

Abstract
Grazing-incidence small-angle scattering (GISAS) is a technique of significant importance for the investigation of thin multilayered films containing nano-sized objects. It provides morphology information averaged over the sample area. However, this averaging together with multiple reflections and the well-known phase problem make the data analysis challenging and time consuming. In the present paper we show that densely connected neural networks (DenseNets) can be applied for GISAS data analysis and deliver fast and plausible results. The extraction of the rotational distributions of hexagonal nanoparticle arrangements is taken as a case study.

1. Introduction

The analysis of grazing-incidence small-angle scattering (GISAS) data is a clear example of the scientific inverse problem. The x-ray or neutron beam hits the sample under grazing incidence angle and gets scattered. The amplitude of the scattered wave is measured by a detector and represents a scattering pattern. The latter one has to be analyzed to obtain information about the sample morphology. Since the phase information is lost, inverse Fourier transformation is impossible. Typical data analysis takes days for a single GISAS pattern. Usually, the following workflow is applied. A scientist builds a sample model, runs a simulation of the scattering process and compares the result with the experimentally observed pattern. This is then repeated many times, while adjusting some model parameters. The latter process is usually automated in modern software frameworks. However, running a simulation for each optimization step is time-consuming and requires considerable computational power.

Magnetic nanoparticle arrangements have received a high level of attention and have a broad range of industrial applications [1–3]. Thus, an exact understanding of their layout is of high importance. Imaging techniques like scanning electron microscopy (SEM) reveal that although the particles are perfectly arranged in an hexagonal lattice, there is some film polycrystallinity present: the film consists of multiple ordered domains rotated with respect to each other. Rotational distribution of these domains is crucial for characterization of the film morphology and for assessment of the film deposition technique. These rotations cause the appearance of additional peaks on the GISAS pattern as shown in the figure 1 and, as any kind of disorder, contribute to diffuse scattering. By analyzing the positions of these peaks, expert scientists can make a good guess about the different angles of rotations. However, extracting the whole rotational distribution is beyond the capability of the human scientist. The left bottom image in the figure 1 gives an impression of complexity of this task: each rotation will contribute not only to the peak position, but also to the relative intensity of the peak and to diffuse scattering. If we limit the amount of possible rotation angles by 120, i.e., $\xi = 0.0^\circ, 0.5^\circ, 1.0^\circ, ..., 59.5^\circ$, it will add 120 new fit parameters to the model. The fitting time of the GISAS pattern will explode.

In the present paper we apply a densely connected convolutional network [5], namely DenseNet169, to extract rotational distributions of hexagonal nanoparticle arrangements from the GISAS patterns.

© 2021 The Author(s). Published by IOP Publishing Ltd
approach overcomes the challenges of the data analysis and allows for obtaining information which would not be possible to obtain otherwise. Since rotational distribution is impossible to fit, there is no way to get labeled experimental data. Therefore, synthetic GISAS patterns generated by BornAgain software have been used to train the neural network. Data augmentation has been applied to the training and validation data to make them as close as possible to experimental ones. Such approach (synthetic data + augmentation) is quite common: see, for example, [6, 7]. In addition, we pay attention to a comprehensive evaluation of the achieved results to ensure that the predictions of the neural network are based on meaningful properties of the input GISAS patterns.

Deep convolutional neural networks have shown high performance for natural image analysis [8] as well as for inverse problem tasks [6, 7, 9–20]. However, to our knowledge, extraction of distributions from the GISAS patterns using deep neural networks, has not been performed so far.

It is important to mention, that the application domain of the presented approach is not limited to the considered use case. Since a GISAS pattern usually contains sample morphology information averaged over the sample surface, any kind of disorder, e.g. size distribution of nanoparticles, rotational distribution of non-symmetric nanoparticles or different shapes of nanoparticles shifts and smears out the observed peaks and contributes to the diffuse scattering and thus increases the complexity of the data analysis dramatically. To our opinion, the developed approach can be applied for extraction of other distributions like shape, size or angle of rotation of nanoparticles, from GISAS patterns. This approach can also be extended to be applied for other neutron and x-ray scattering techniques where extraction of distributions is desired.

2. Related work

Complexity of GISAS data analysis makes it attractive for application of deep learning. Therefore a number of works is devoted to various aspects of GISAS data analysis can be found. For example, the works [18] applies self-designed CNN for classification of the GISAXS patterns according to the shape of the nanoparticles. This helps with selection of model for further data analysis.

The work [15] successfully applies modified AlexNet architecture to simulated and measured GISAXS patterns for classification of seven kinds of lattice orientation of 3D nanoparticle arrangements (cubic 100, cubic 110, bcc 100, bcc 110, fcc 100, fcc 111, hcp 0001). To distinguish between different kinds of the lattice orientation the neural network has to pay attention to positions and intensities of the Bragg peaks. Such classification also decreases the usual data analysis time by suggesting the model for particular lattice orientation.
The neural network in the present paper addresses much more complex physical problem, since rotational distribution of the nanoparticle domains smears out the Bragg peaks and contributes also to diffuse scattering observed in the GISAS pattern. Moreover, the collected information is averaged over a large area of the sample surface. Thus, more complex neural network architecture is required. Here we consider the extraction from the GISAS pattern of a 120-dimensional vector of real numbers as a regression problem, although it might seem to be similar to a classification task with 120 classes.

Similarly, the work [7] addresses the regression problem by reconstructing the quaternion of rotation and size of icosahedron nanoparticles from wide-angle diffraction patterns. In contrast to wide-angle diffraction, the GISAS technique usually collects sample morphology information averaged over the sample surface. Therefore, in the present work, by extracting rotational distributions from the GISAS patterns we suggest a solution for a significantly more complex physical problem.

3. Training a deep neural network

While designing the workflow, both empirical knowledge and general recommendations [21, 22] have been considered. The sketch in figure 2 shows the workflow used in the present work. First, synthetic labeled data are generated using the BornAgain software [4]. In the figure they are shown as a GISAS pattern and a yellow rectangle named reference Y. Next, data augmentation is applied. This step is shown as a noisy GISAS pattern in the figure 2. Then, the augmented data are used as an input for the deep neural network (blue rectangle marked as DNN) which generates a prediction (green rectangle marked as predicted Y). Then, the loss function is computed for both, training and validation data, and parameters of the neural network are adjusted in order to minimize the loss. This process is then repeated for about 200 epochs until an acceptable loss value has been reached. The details on each step are presented below.

3.1. Data generation

The main challenges in generating lots of data are performance and possible bias. For the present work $5 \times 10^5$ training and $5 \times 10^4$ validation datasets have been generated. Each dataset consists of a GISAS pattern represented as a matrix of size $256 \times 256$ and a lattice rotation distribution represented as a vector of size 120. Each element of this vector corresponds to the probability of a lattice rotation at a particular angle. In the present work, the rotation angles range from $0^\circ$ to $60^\circ$ with a step of $0.5^\circ$.

To save computational resources, a simple sample model of spherical nanoparticles arranged in a 2D hexagonal lattice has been chosen. Example real space views of such a sample model are shown as insets in figure 1. The number of different lattice rotation angles present in each GISAS pattern has been varied from 1 to 5.

To prevent overfitting to a particular sample or experiment parameter, these parameters have been varied during the data generation. The full list of parameters and their ranges is presented in the table 1. Here, the lattice constant, decay length and position variance parameters describe the 2D hexagonal lattice; particle radius characterizes the size of the nanoparticle; finally, beam wavelength and beam inclination describe properties of the x-ray beam. For performance reasons, such experimental characteristics as angular beam divergence, wavelength distribution, detector resolution, masked areas and background signal have been ignored during the data generation step.

3.2. Data augmentation

Randomness present in the generated data is still not sufficient for a plausible performance of the neural network. Moreover, the trained neural network model should also make reasonable predictions for
experimental GISAS patterns. These circumstances make it necessary to introduce additional randomness in the training batches. Thus, the following empirically chosen data augmentation steps have been consequently applied to both training and validation data:

1. Intensity scaling as $I_{\text{scaled}} = I \cdot 10^n$, where $n$ is a random number in the range $[-2, 2]$ sampled from a uniform distribution.

2. Poisson noise.

3. Beam stop: intensities in a rectangular area of a random width in the range from 6 to 10 pixels and of a random height in the range from 80 to 120 pixels at a random position close to the center of the GISAS pattern are set to zero.

4. Detector mask: intensities in a vertical stripe of a random width in the range from 3 to 5 pixels at a random position within the GISAS pattern are set to zero.

5. Logarithm as $I_{\text{new}} = \log(I + \varepsilon)$, where $\varepsilon = 0.1$.

6. Random crop, e.g. GISAS pattern of size $256 \times 256$ pixels is randomly cropped to the size of $224 \times 224$.

7. Normalization of intensities to a zero mean and unity standard deviation (standardization), e.g. $I_{\text{new}} = \frac{I - \bar{I}}{\sigma}$, where $\bar{I}$ is a mean intensity value and $\sigma$ is its standard deviation.

Intensity scaling is important to make the neural network model invariant to particular values of the absolute intensities. Poisson noise, beam stop, detector mask and random crop are necessary to account for the features present in experimental data. They also introduce additional random distortions important for a better performance of the neural network. Since usually GISAS intensities vary over a few orders of magnitude, a logarithmic transformation is needed to reduce this variation. Normalization of intensities is crucial for the convergence of the gradient-based optimization algorithm. It is important to apply the mentioned data augmentation steps in the given order. Therefore, some simple steps, like logarithm, have not been applied to the training data in advance.

These data augmentation steps require very little computation power in comparison to the data generation step. Hence, they can be applied during the training of the neural network. This also implies that the training data used in each epoch will be slightly different. This reduces the overfitting that could occur by using a static training data set.

### Table 1. Parameters varied during data generation.

| Parameter            | Min value | Max value | Units |
|----------------------|-----------|-----------|-------|
| lattice constant     | 30        | 40        | nm    |
| decay length         | 100       | 250       | nm    |
| position variance    | 0         | 15        | nm$^2$|
| particle radius      | 8         | 13        | nm    |
| beam wavelength      | 1.2       | 1.5       | Å     |
| beam inclination     | 0.15      | 0.18      | °     |

#### 3.3. Neural network architecture

For the present work a densely connected convolutional neural network (DenseNet) [5] architecture has been chosen. The peculiarity of this architecture is that it connects each layer to every other layer in a feed-forward fashion. This means that the feature maps used in each layer as an input propagate to all subsequent layers. The main advantages of the DenseNet architecture are that it alleviates the vanishing gradient problem, strengthens feature propagation and feature reuse, and has a rather small number of parameters [5]. The latter is especially important for neutron and x-ray scattering data due to the limited amount of training data, since even generation of synthetic datasets can be time-consuming.

During this work, training of DenseNet121 (7 millions of trainable parameters) and DenseNet169 (12.7 millions of trainable parameters) architectures has been performed. Both DenseNet121 and DenseNet169 have shown a comparable performance. However, the validation loss achieved with Densenet169 was slightly (by about 10%) lower. Thus, all further results presented in this paper are computed using the trained DenseNet169.
3.4 Loss function

The loss metric used during the training of the neural network is the Kullback-Leibler divergence between the label, i.e. the distribution of orientations used to generate the scattering image, and its prediction. The Kullback-Leibler divergence (later mentioned as KL divergence or KLD) between two distributions \( y \) (label) and \( \hat{y} \) (prediction) is computed in the following way [23, 24]:

\[
\text{KL}(y\|\hat{y}) = \sum_{\xi} y(\xi) \log \frac{y(\xi)}{\hat{y}(\xi)}
\]

where \( \xi \) is the lattice rotation angle.

While this loss function is often used in classification tasks, it receives a more clear interpretation in our case, which is a regression task for distributions. From the perspective of information theory, the Kullback-Leibler divergence encodes the amount of information lost by using the predicted distribution instead of the real distribution (i.e. the one used for generating the data). As we will see later on, our setting also allows for a quantitative interpretation of the loss, by comparing its value with the one that we would get from a uniform distribution.

3.5 Minimization strategy

For minimization, a stochastic gradient descent (SGD) algorithm with momentum equal to 0.9, clipnorm equal to 10 and adaptive learning rate has been used. Momentum determines the relative contribution of the gradients from earlier training steps to the neural network parameters’ change. The parameter clipnorm addresses the exploding gradients problem by clipping the gradient value. The learning rate has been adjusted as follows. First, during the first warm-up steps \( N_{\text{warm-up steps}} \), the new learning rate \( \eta_k \) for a training step number with number \( k \) is computed as

\[
\eta_k = \eta_0 \cdot \left(1 - \frac{k}{N_{\text{total}}}\right)^d \cdot \frac{k}{N_{\text{warm-up steps}}}
\]

where \( \eta_0 = 0.01 \) is the base learning rate, polynomial degree \( d = 1 \), and \( N_{\text{total}} \) is the total number of steps computed as a number of epochs multiplied by number of steps per epoch. For 200 epochs and batch size of 16, \( N_{\text{total}} = 6.25 \times 10^6 \).

Figure 3 shows how the learning rate \( \eta \) changes with training step number \( k \). During the warm-up phase \( k < N_{\text{warm-up steps}} = 1000 \), the learning rate computed according to equation (2) linearly increases up to it’s base value \( \eta_0 \). From then on, \( \eta \) is computed according to (3) and decreases at each step. This decrease is linear, since the polynomial degree \( d \) is set to 1.

Such a strategy accounts for a better exploration of the neural network parameters space during the warm-up phase and for more accurate, in small steps, convergence to a minimum of the loss function in the later phase of the training.

4. Evaluation of the trained network

Since the ultimate goal of the project is to have a trained neural network that can identify the presence of different lattice orientations from scattering data in a fast and reliable way, the trained network’s performance should ideally be evaluated by measuring how well it performs on a reasonably large set of experimental data,
with known properties (in this case, known distributions of lattice orientations). Besides the question of which metric to use as a meaningful measure of the quality of the network on this data, there is the more evident problem that we do not have such a data set and that it cannot be easily generated. For this reason, we have to resort to different techniques, mostly based on metrics obtained from the validation data.

In this section, a number of different evaluation techniques will be used to visualize and give a qualitative assessment of the trained neural network. As a basis for most of the analyses we generated a flat data file from the validation data and the neural network’s predictions. Each row denotes a single example in the validation data and contains:

- The Kullback-Leibler divergence between the real distribution used and its prediction by the network. This is exactly the loss function used during the training steps.
- The parameter values used during the generation of the validation data, including the distribution of orientations.

The parameters that are randomly assigned and used during the data augmentation step are not recorded in this file.

### 4.1. Training and validation loss

In figure 4, the evolution of the loss function is shown as a function of the training epoch, both for the training and validation data set. The loss function is plotted in natural units of information (nats). As one can see, both train and validation losses are very close to each other, even overlap. This is a sign that the model, despite its complexity, does not overfit the training data and generalizes well on the validation data set. The training curve (black line) decays smoothly and does not contain spikes. Thus, no sign of exploding gradients is present. Light fluctuations of the validation loss (gray line) are highly probably caused by a small batch size. Due to the hardware limitations, the maximal possible batch size was 16.

Figure 5 shows a histogram of the loss on the validation set. In this histogram is shown, that 90% of the losses are smaller than 1.77 and 95% smaller than 2.27.

To get a quantitative impression of the quality of the predictions on the validation set, one can plot the loss for each example together with the entropy of its real distribution. As a measure, we also plot the KL divergence for a maximum entropy guess, which is a uniform distribution. In this case, the KL divergence is equal to \( \log(120) - H(p) \), with \( H(p) \) the entropy of the real distribution. This is done in figure 6. From this plot, one sees that in almost all cases, the prediction is considerably better than the uniform guess. Only for 0.30% of the cases is the prediction worse.

### 4.2. Relation between the inputs and the quality of the network’s prediction

As a way to study the relation of certain model parameters, including the orientational distribution, with the quality of the prediction, one can calculate the pairwise correlation coefficients between the loss values and all other parameter values. Both for positive and negative correlation, the largest coefficients are obtained for the angle probabilities. And even there, the values are quite limited (<7%).
Instead of looking at correlation coefficients between the loss function and each individual input parameter, one can also try to find linear combinations of the input parameters that, in some predefined sense, have a maximal influence on the loss function. Partial least squares analysis (PLS) is one such method that we will use here for the lattice orientation parameters. In PLS, the maximal influence is defined to be a maximal covariance between latent variables of both input and output. Since the loss function consists of a single scalar, only one latent variable can be constructed.

We start by defining the matrix $X$ such that each row denotes the probabilities for the lattice orientation angles for a single example of the validation data set. In our case, this matrix has dimensions $5 \cdot 10^4 \times 120$. The loss vector $Y$ contains the loss function for each prediction on the same data set ($5 \cdot 10^4$ elements). PLS now tries to find the direction in the space of distributions that maximizes the covariance between the loss and the data's component in that direction:

$$u_1 = \text{argmax}_{u_1 \in \mathbb{R}^N} \text{Cov}(X \cdot u_1, Y).$$

In other words, the vector $u_1$ denotes the mode whose presence in the distribution of lattice orientations has the largest covariance with the loss. One can interpret it then as follows: orientation angles where $u_1$ is large are more likely to lead to higher loss than angles where $u_1$ is small. The peaks in $u_1$ thus denote orientations whose presence is difficult to predict, whereas the valleys denote angles that are easier to predict.

Figure 7 shows the solution vector $u_1$ for the partial least squares method (black solid line). As would be expected for a network that correctly identifies certain peaks in reciprocal space with specific lattice orientations, the lower areas of $u_1$ coincide with orientations that cause specific peaks to appear in the detector. For a hexagonal lattice, the correspondence between the orientation angle of the lattice and the Miller index for the...
relevant peaks is as shown in table 5. Lattice rotation angles corresponding to these miller indexes are denoted in the figure 7 with red dashed lines.

To ensure that this effect is not caused by bias of the training data, we have additionally trained DenseNet169 on a smaller dataset with up to 10 non-zero distribution components. The result of the PLS analysis for this data is shown in the figure 7 as a gray dashed line. As one can see, minima of the solution vector \( u_1 \) correspond to the same Miller indices for both datasets.

The orientations that are harder to predict correspond to regions where no reciprocal lattice peak directly crosses the detector image. For these orientations, the peaks fall outside of the detector image (the spacing between lattice planes is too small). This provides a strong indication that the trained network has an internal representation of the relation between lattice orientations and reciprocal lattice peaks.

### 4.3. Analysis of possible bias

One of the main concerns in the training of a deep neural network is the presence of bias in the training data. Although we applied a number of methods to minimize this, like data augmentation and normalization, there still remains a clear bias in the data: for performance reasons, only distributions with a maximum of five non-zero probabilities were used in the training (and validation) data.

It is therefore crucial to at least get a qualitative understanding of how this bias might possibly influence predictions on real experimental data. However, since we do not have labeled data from experiments, we will assess the quality by looking at how the network performs on types of distribution that are clearly outside the trained data set. First, we generate a dataset with six to ten non-zero probabilities and assess the quality of prediction by plotting the loss for each example together with the entropy of its real distribution. The result is shown in the figure 8. Although the quality of prediction for these data is definitely worse than for data with a maximum of five non-zero probabilities (see figure 6 for comparison), it is still reasonable: only for 1.79% of the cases is the prediction worse than the uniform guess.

Second, we assess a prediction for a uniform distribution. An example GISAS pattern for uniformly distributed lattice orientations is shown in the figure 1.

In figure 9(a), a scatter plot is shown of 100 different predictions for a uniform example. To generate such amount of predictions for a single GISAS pattern, data augmentation steps, the same as for training and validation data, have been applied. Points indicate the mean value of the predicted probability for each rotation angle and the error bars a standard deviation. Although the predictions show some tendency towards certain angle regions, the overall picture is quite noisy and a standard deviation is high.

To find out whether this effect is fundamental, we have trained a DenseNet169 on a smaller dataset containing up to ten non-zero probabilities. The prediction made by this neural network for a uniformly distributed rotations is shown in the figure 9(b). As one can see, it looks significantly smoother than prediction

---

**Table 2.** Miller indexes.

| \( \xi \)   | 0° | 19.1° | 30° | 40.9° | 60° |
|-------------|----|-------|-----|-------|-----|
| Miller index| (0, 1) | (2, 1) | (1, 1) | (1, 2) | (1, 0) |

---

**Figure 7.** Solution of the partial least squares method. \( \xi \) is the lattice rotation angle.
shown in the figure 9(a) and its standard deviation is also smaller. This allows for a suggestion, that presence of larger amount of non-zero probabilities in the training data would eliminate the bias. However, generation of such a dataset is very resource-consuming. Thus, it is important to consider the bias while analyzing distributions predicted for experimental data.

4.4. Prediction for experimental data

For evaluation of the DenseNet169 performance for experimental data, three GISAXS patterns measured at the Jülich in-house instrument GALAXI [25] for different kinds of 2D hexagonal nanoparticle arrangements have been taken. Further on, we mention these data as Experiment 1 [26], Experiment 2 [27], and Experiment 3 [28]. GISAXS patterns for experiments 1 and 3 look quite similar, although the pattern for experiment 3 is noticeably more noisy. Both patterns clearly show peaks corresponding to the Miller indices (1, 1). Thus, a possible guess of the expert scientist for these 2 patterns is a rotational distribution with non-zero probabilities only for two angles: $\xi = 0^\circ$ and $\xi = 30^\circ$. In contrast, positions, relative intensities and number of observed peaks in the GISAXS pattern for experiment 2 are quite similar to those of the uniformly distributed $\xi$ shown in the figure 1. Thus, suggestion of the expert scientist for experiment 2 is an absence of any preferred lattice orientation, i.e. uniform distribution.

Figure 10 presents a comparison of distributions predicted by DenseNet169 (scatter plots) to ones suggested by an expert scientist (given in a top-left corner of each plot). To assess the stability of predictions, the following data augmentation has been applied to experimental GISAS patterns. First, the image of size $1043 \times 981$ pixels (size of GALAXI detector) has been cropped to a reasonable region of interest, where the main part of the signal...
is observed. Then, a rebinning to size 256 × 256 pixels has been applied. After that, random distortions like beam stop and random crop have been applied to get 100 slightly different GISAS patterns for each experimental one. Logarithm and normalization have been applied to each pattern as well.

As one can see in the figure 10, for the experiment 1 DenseNet169 makes a prediction very similar to the one of the expert scientist. Although there are a few more non-zero probabilities present, the predicted probabilities for lattice rotation angles $\xi = 0^\circ$ and $\xi = 30^\circ$ are significantly larger than for other $\xi$ values. Predictions made by DenseNet169 for experiments 2 and 3 look quite similar to one made for a uniform distribution (bottom plot): they contain a lot of non-zero probabilities, the maximal predicted probability is rather small and the overall picture looks noisy. To make a more clear statement, let’s compare the statistical characteristics of all 4 predicted rotational distributions.

Figure 11 shows the boxplots: a standardized way to plot the statistical characteristics of distributions. Boxes show the interquartile range of the predicted probabilities (IQR, 25th to 75th percentile), whiskers indicate ‘minimum’ (IQR − 1.5 · Q1) and ‘maximum’ (IQR + 1.5 · Q3), where Q1 and Q3 are the 25th and 75th percentiles respectively. Points show the ‘outliers’. As is seen from the figure 11, statistical characteristics of predictions made by DenseNet169 for experiment 2, experiment 3 and a simulated GISAS pattern with a uniform orientational distribution look rather similar, while being significantly different for experiment 1.

To take a final decision, one can compute the KLD between predicted and uniform distribution for each experiment. This will show how much information will get lost if we substitute the predicted distribution with the uniform one. The result is presented in the figure 12 as black dots with error bars. Since KLD is not constrained from the top, we need to compare the computed values to some reasonable measure. Here we take the validation loss (mean value is shown as a blue solid line in the figure 12) as such a measure and take the 90th percentile (blue dashed line) as a threshold. As one can see, KLD for experiments 2 and 3, as well as for uniform lattice orientations are far below the threshold, while KLD for experiment 1 is, in contrast, far above. Based on this, it is reasonable to assume that DenseNet169 predicts for experiments 2 and 3 a uniform rotational distribution.
This conclusion agrees with an expert scientist’s prediction for experiment 2. However, for the experiment 3, the rotational distribution suggested by expert scientist contains only two rotations ($\xi = 0^\circ$ and $\xi = 30^\circ$) with equal probabilities of 0.5. It is clear, that for this experiment the neural network prediction is very different from one made by an expert scientist. Since the true distributions for experimental data are not known, we do not have a direct way to test the correctness of the prediction made by DenseNet169. However, an indirect check is possible: we can simulate a GISAS pattern with predicted and uniform orientational distributions and compare them to the measured one. Such comparison is shown for experiment 3 in the figure 13. The plot on the left displays 2D GISAXS pattern for experiment 3 simulated with BornAgain software (left half) and the measured one (right half). The plot in the right part shows a slice along $Q_y$ on the $Q_z$ level of the Yoneda peak (indicated as a thin dashed line on the 2D GISAXS pattern). As one can see, the uniform distribution (red dashed line) reproduces the experimental data (black dots) better than the expert scientist’s guess (gray solid line). Although both rotational distributions match the peak positions and intensities, the diffuse scattering is better reproduced by a uniform rotational distribution.

To strengthen our interpretation of the distribution predicted by DenseNet169 as uniform, we have performed also a simulation of the GISAXS pattern with ‘as predicted’ distribution (shown in the figure 10). The result is represented by a light blue line in the figure 13. Although this distribution well reproduces diffuse scattering, peak shapes and relative intensities are better reproduced by the uniform distribution.

It can be noted in figure 10, that in contrast to the prediction for the uniform distribution, predictions for Experiments 2 and 3 do not show higher probabilities for the angles mentioned in the table 5. The possible reason for that may be that some effects naturally present in the experimental data, e.g. background signal, angular beam divergence, wavelength distribution, detector resolution, were not present in the training data, but may have influence on the prediction. Since these effects mainly contribute to the broadening of observed peaks and signal-to-noise ratio of the GISAS pattern, we suppose that their influence on predictions for highly-disordered systems (like experiments 2 and 3) should be high, while for less disordered systems like Experiment 1 relatively low.
4.5. Saliency maps
As a final sanity check, it is reasonable to plot saliency maps or attention maps. Technically, such a map is a gradient of the prediction score with respect to the image [29]. Thus, it can be interpreted as a tool to highlight image regions which contribute most to the prediction.

Maps presented in the figure 14 have been produced with keras-vis [30]. For this figure, predictions made by DenseNet169 for experiment 1 have been chosen. It is not shown, but the whole GISAS pattern contributes to the prediction of the lattice rotation angle. This means, that the neural network pays attention also to diffuse scattering. The maximal gradients are observed in the range of higher scattering intensities. As one can see in the figure 14, the neural network pays attention to the structural peaks and the shape of the map is different for prediction of different lattice rotation angles. This confirms the results of the PLS analysis.

5. Conclusions and outlook
In the present work we have applied a DenseNet neural network to predict rotational distributions for GISAS patterns of hexagonally arranged nanoparticles. Although the neural network has been trained on synthetic data produced by BornAgain software [4], we have shown that it makes also reasonable predictions for experimental data. Analysis of the trained neural network shows, that it has learned an internal representation of the relation between lattice orientations and reciprocal lattice peaks.

The results achieved are of high importance for neutron and x-ray communities. They help in understanding of the measured GISAS patterns and accelerate the data analysis, since the trained neural network can provide in milliseconds a reasonable guess of a rotational distribution, which is usually not possible to fit. Although the true distribution is as a rule not known, the quality of prediction can always be assessed by comparison of simulated pattern with experimentally observed one. Since deep learning science continuously develops and new, more effective, neural network architectures are discovered, this work can be definitely improved in the future. Further on, the presented approach can be extended for other data analysis tasks and scattering techniques where a prediction of distributions is required.

Acknowledgments

The authors would like to thank BornAgain team, especially Dr. Gennady Pospelov, for the kind support during this work. Authors are also grateful to Dr. Li-Ming Wang, Asmaa Qdemat and Dr. Emmanuel Kentzinger for provided GISAXS data.

Data availability statement

The data that support the findings of this study are available upon reasonable request from the authors. https://github.com/mganeva/ba-learning.
Appendix. Soft- and Hardware

All the code for the present work has been written in Python. Following packages have been used: TensorFlow [31], Keras [32], Pandas [33], NumPy [34, 35], matplotlib [36] and seaborn [37]. This work has been performed using office desktop computers with AMD Ryzen 7 1800x Eight-Core CPU, 64 Gb of RAM and Nvidia GeForce GTX 1080 GPU with 8 Gb of GPU memory.

ORCID iDs

Walter Van Herck  
https://orcid.org/0000-0001-7958-3330

Marina Ganeva  
https://orcid.org/0000-0001-7624-101X

References

[1] Nie Z, Petukhova A and Kumacheva E 2010 Nat. Nanotechnol. 5 15
[2] Velev O D and Gupta S 2009 Adv. Mater. 21 1897
[3] Ukleev V, Khassanov A, Snigireva I, Kononov O and Vorobiev A 2016 Thin Solid Films 616 43
[4] Pospelov G, Van Herck W, Burle J, Carmona Loaiza J M, Durniak C, Fisher J M, Geneva M, Yurov D and Wuttke J 2020 J. Appl. Crystallogr. 53 262
[5] Huang G, Liu Z, Maaten L V D and Weinberger K Q 2017 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (https://doi.org/10.1109/cvpr.2017.243)
[6] Laanait N, He Q and Borisevich A Y 2019 Reconstruction of 3-d Atomic Distortions from Electron Microscopy with Deep Learning (arXiv:1902.06876)
[7] Stelow T, Schmidt R, Peltz C, Fennel T and Scheel S 2019 Fast Reconstruction of Single-Photon Diffraction Images Through Deep Learning (arXiv:1906.08685)
[8] He K, Zhang X, Ren S and Sun J 2016 2016 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (https://doi.org/10.1109/cvpr.2016.90)
[9] Ushizima D, Yang C, Venkatakrishnan S, Araujo F, Silva R, Tang H, Mascalenshas J V, Hexemer A, Parkinson D and Seahian J 2016 2016 Applied Imagery Pattern Recognition Workshop (AIPR) (IEEE) pp 1–12
[10] Wang B, Yager K, Yu D and Hoai M 2017 2017 IEEE Winter Conf. on Applications of Computer Vision (WACV) (IEEE) pp 697–704
[11] Meister N, Guan Z, Wang J, Lashley R, Liu J, Lhermitte J, Yager K, Qin H, Sun B and Yu D 2017 2017 New York Scientific Data Summit (NYSDS) (Piscataway, NJ: IEEE) pp 1–6
[12] Park W B, Chung J, Jung S, Sohn K, Singh S P, Pyo M, Shin N and Sohn K-S 2017 IUCrJ 4 486
[13] Araujo F H, Silva R R, Medeiros F N, Parkinson D D, Hexemer A, Carneiro C M and Ushizima D M 2018 Expert Syst. Appl. 109 35
[14] Liu J and Yager K G 2018 IUCrJ 5 737
[15] Liu S, Melton C N, Venkatakrishnan S, Pandolfi R J, Freychet G, Kumar D, Tang H, Hexemer A and Ushizima D M 2019 MRS Communications 9 586
[16] Liu S, Li J, Bennett K C, Ganoe B, Stauch T, Head-Gordon M, Hexemer A, Ushizima D and Head-Gordon T 2019 The Journal of Physical Chemistry Letters 10 4558 PMID: 31305081
[17] Laanait N, Romero J, Yin J, Young M T, Treichler S, Starkenchos V, Borisevich A, Sergeev A and Matheson M 2019 Exsascale Deep Learning for Scientific Inverse Problems (arXiv:1909.11530)
[18] Ikemoto H, Yamamoto K, Tsyuma H, Yamashita D, Nakamura M and Okuda H 2020 J. Synchrotron Radiat. 27 1069
[19] Guan Z, Qin H, Yager K, Choo Y and Yu D 2018 British Machine Vision Conference 0828 1–10 (http://bmvc2018.org/contents/papers/0828.pdf)
[20] He H, Liu C and Liu H 2020 Science 370 10906
[21] Goodfellow I, Bengio Y and Courville A 2016 Deep Learning (Cambridge, MA: MIT Press) (http://deeplearningbook.org)
[22] Shorten C and Khoshgoftaar T M 2019 Journal of Big Data 6 60
[23] Kullback S and Leibler R A 1951 Ann. Math. Statist. 22 79
[24] MacKay D J C 2003 Information Theory, Inference and Learning Algorithms (Cambridge: Cambridge University Press)
[25] Kzentzinger E, Krutyeva M and Rückner U 2016 Journal of Large-Scale Research Facilities JLSRF 2 A61
[26] Qdemat A, Kzentzinger E, Xu J, Portale G, Rückner U, Geneva M, Yurov D, Mattauch S and Brückel T 2d Ordered Arrays of Ferrimagnetic Cobalt Ferrite Nanodots: X-ray and Polarized Neutron Reflectivity (in preparation)
[27] Qdemat A, Kzentzinger E, Buitenhuys J, Rückner U, Geneva M and Brückel T 2020 RSC Adv. 10 18383
[28] Wang L M 2018 Dissertation RWTH Aachen University published at Publicationserver of RWTH Aachen University, (https://doi.org/10.18154/RWTH-2018-22451
[29] Simonyan K, Vedaldi A and Zisserman A 2013 Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps (arXiv:1312.6034)
[30] Kotikalapudi R and (contributors) 2017 keras-vis (https://github.com/raghakot/keras-vis)
[31] Abadi M et al 2015 TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems software available from tensorflown.org, URL (https://tensazorflow.org/)
[32] Chollet F et al 2015 Keras (https://keras.io)
[33] McKinney W et al 2010 IPython in Science Conf. 445 (Austin, TX) 56–61 (http://conference.scipy.org/proceedings/scipy/2010/pdfs/mckinney.pdf)
[34] Oliphant T E 2006 A Guide to NumPy vol 1 (USA: Trelgol Publishing) (https://web.mit.edu/dvp/Public/numpybook.pdf)
[35] van der Walt S, Colbert S C and Varoquaux G 2011 Computing in Science Engineering 13 22
[36] Hunter J D 2007 Comput. Sci. Eng. 9 90
[37] Waskom M et al 2020 mmaskose/seaborn: v0.10.0 (January 2020) (https://doi.org/10.5281/zenodo.3629446)