Classification of diffraction patterns in single particle imaging experiments performed at x-ray free-electron lasers using a convolutional neural network

IOP Publishing Mach. Learn.: Sci. Technol. 2 (2021) 025014 https://doi.org/10.1088/2632-2153/abd916

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
14 August 2020

REVISED
6 November 2020

ACCEPTED FOR PUBLICATION
6 January 2021

PUBLISHED
26 February 2021

PAPER

Classification of diffraction patterns in single particle imaging experiments performed at x-ray free-electron lasers using a convolutional neural network

Alexandr Ignatenko[1], Dameli Assalauova[1], Sergey A Bobkov[2], Luca Gelisio[1], Anton B Teslyuk[2,3], Viacheslav A Ilyin[2,3,4] and Ivan A Vartanyants[1,5]

1 Deutsches Elektronen-Synchrotron DESY, Notkestrasse 85, 22607 Hamburg, Germany
2 National Research Centre 'Kurchatov Institute', pl. Akademika Kurchatova 1, 123182 Moscow, Russia
3 Moscow Institute of Physics and Technology, Institutskiy per. 9, 141701 Dolgoprudny, Moscow region, Russia
4 ITMO University, Kronverksky Pr. 49, 197101 St. Petersburg, Russia
5 National Research Nuclear University MEPhI, Kashirshkoe sh. 31, 115409 Moscow, Russia

E-mail: Ivan.Vartaniants@desy.de

Keywords: single particle imaging, classification, convolutional neural network, transfer learning

Abstract

Single particle imaging (SPI) is a promising method of native structure determination, which has undergone fast progress with the development of x-ray free-electron lasers. Large amounts of data are collected during SPI experiments, driving the need for automated data analysis. The necessary data analysis pipeline has a number of steps including binary object classification (single versus non-single hits). Classification and object detection are areas where deep neural networks currently outperform other approaches. In this work, we use the fast object detector networks YOLOv2 and YOLOv3. By exploiting transfer learning, a moderate amount of data is sufficient to train the neural network. We demonstrate here that a convolutional neural network can be successfully used to classify data from SPI experiments. We compare the results of classification for the two different networks, with different depth and architecture, by applying them to the same SPI data with different data representation. The best results are obtained for diffracted intensity represented by color images on a linear scale using YOLOv2 for classification. It shows an accuracy of about 95% with precision and recall of about 50% and 60%, respectively, in comparison to manual data classification.

1. Introduction

Single particle imaging (SPI) enables structural investigations of non-crystalline objects [1, 2]. Many particles with random orientation are illuminated by an x-ray free-electron laser (XFEL) beam, which consists of intense pulses with durations of tens of femtoseconds [3–5]. Scattering patterns of the investigated particles are therefore collected before radiation damage takes place, in a so-called 'diffraction before destruction' [6] experiment. Since SPI does not require crystallization, it enables investigation of objects that are hard—or impossible—to crystallize, like viruses, proteins, organelles, and others [7–12].

A general analysis pipeline for SPI data consists of several steps as first proposed in [2] and further extended in [11, 12]. One of the important steps is the classification of all diffraction patterns measured in an XFEL experiment, and, specifically, identification of single hits, i.e. events containing the scattering pattern of a single particle. Earlier this task was performed by different methods, such as principal component analysis or support vector machine [13], as well as using the expectation-maximization (EM) algorithm [12] developed in cryogenic electron microscopy [14]. In this work, we propose to select single hits from experimental data set using a convolutional neural network (CNN) approach. The benefits of the CNN over EM are its short computation time and inherent parallelism that enable it to be used for quasi-online classification. CNNs are known to give excellent results for the tasks of image classification and object detection [15, 16]. CNN-based solutions have been successfully applied recently to classify diffraction

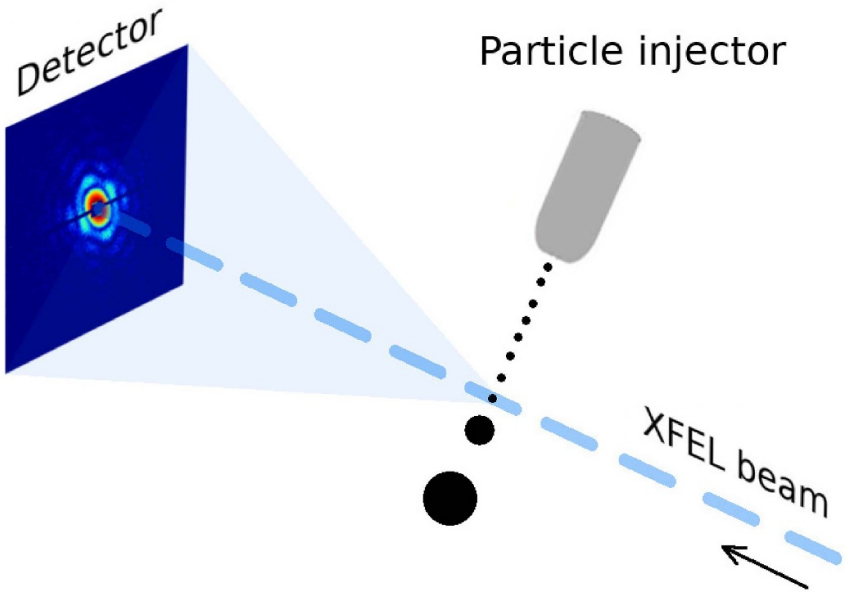

Figure 1. Schematic representation of an SPI experiment. Single viruses or other particles are injected into the focused XFEL beam. A large panel detector is employed to record the scattered intensity.

patterns collected in coherent diffraction imaging experiments at XFELs [17, 18] and tomography experiments at synchrotron sources [19].

Here, we present the results of classification of diffraction patterns in SPI experiments using a CNN. We use transfer learning to train the network for classification with a limited amount of training and validation data, and demonstrate that this approach is comparable with the EM-based selection method.

2. Methods

2.1. Experiment and data preparation

The SPI experiment (see figure 1) was performed using the atomic molecular optics (AMO) instrument [20, 21] at the Linac Coherent Light Source at SLAC National Accelerator Laboratory in the frame of the SPI initiative [22] (experiment AMOX34117 [23]). Samples of PR772 bacteriophage [23, 24] were aerosolized using a gas dynamic virtual nozzle in a helium environment [25]. The particles were injected into the sample chamber using an aerodynamic lens injector [8, 26]. The samples in the particle stream intersected the focused and pulsed XFEL beam. The XFEL had a repetition rate of 120 Hz, an average pulse energy of ∼2 mJ, a focus size of ∼1.5 μm, and a photon energy of 1.7 keV (wavelength 0.729 nm). Diffraction patterns were recorded by a pn-type charge coupled device (pnCCD) detector [27,] mounted at 0.130 m distance from the interaction region. The size of the panel was 512 by 1024 pixels with a pixel size of 75×75 μm^2.

The total amount of experimental data collected was approximately 1.2×10^9 patterns [23]. Only parts of the data that were classified as hits contained scattering signals in the diffraction pattern. The hit finding was performed using the software 'psocake' in the 'psana' framework [28]. As a result, about 1.9×10^7 diffraction patterns were selected as hits from the initial set of experimental data [23]. Prior to classification, the center of each diffraction pattern was determined and a mask was introduced to remove bad pixels and areas of the detector with high background and saturation [12]. The scattering signal near the center of the diffraction pattern was considered to be background and subtracted from each diffraction pattern. Particle size filtering was performed with a particle size range from 55 nm to 84 nm being selected as described in [12]. As an outcome of this size filtering, 18 213 patterns were selected for single hit classification. In this work a single hit is defined as a diffraction pattern corresponding to x-ray scattering on a single particle and is distinguished from a multiple hit that originates from x-ray scattering on a few particles or scattering from other objects (for example, water droplets). The goal of this work is to perform binary classification of measured data and to sort diffraction patterns belonging to a 'single hit' class from other patterns.

In our previous work [12], we used an EM-based classification method of an iterative unsupervised clustering algorithm with a predefined number of clusters. The algorithm starts from a random model for each cluster. At each iteration the cluster model is compared with the 2D rotation of each diffraction pattern and the probability to belong to a certain class is calculated. After evaluation of the probabilities, a new model

for each cluster is calculated by weighted averaging of all patterns. The weights are defined by the computed probabilities. When the EM algorithm converges, the clusters that correspond to the same class are selected manually by an expert. As a result of this EM classification, 1085 single-particle patterns were identified.

The manual selection performed by the visual inspection of the patterns was considered as a ground truth [23]. The number of single hits in the case of manual selection was 1393. From this selection, 1196 single hits belonged to a data set filtered by a particle size in the range from 55 nm to 84 nm and were considered as a ground truth for EM classification. Visual inspection of the manual selection did not reveal any false positive patterns.

2.2. CNN choice

The choice of a CNN for our task was based on the consideration that binary classification of single hits for the SPI experiments is different from the classification of features for the diffraction patterns in the experiments described in [18]. Indeed, binary classification is easier than having multiple cases. On the other hand, diffraction patterns in SPI experiments contain scattering signal from particles in many arbitrary orientations. It is therefore more difficult for a CNN to learn that all diffraction patterns contain an image of the same particle, and to distinguish them from all other possible kinds of diffraction patterns.

In this work we used a fast object detector YOLOv2 [29] from the open source neural network framework Darknet [30] to perform classification. Its relatively shallow CNN is similar to the one used in [18]. We compared the performance of YOLOv2 with the more recent and deeper version YOLOv3 [31]. This deeper network was considered in order to understand whether a significant gain in performance may be obtained between these two networks. This could also justify an increase in complexity of calculations by using the YOLOv3 network. Importantly, both YOLO versions, being real-time object detection systems, have shorter computational time and at the same time give similar performance in comparison to other detection methods [29, 31].

2.3. CNN description

CNNs consist of layers, the core building blocks being convolutional layers. Together with other layers, such as pooling layers, they form a network [32]. The layers are characterized by a set of filters or kernels, containing weights. In a convolutional layer the convolutions of the input with different filters are computed. Different features that can appear at any point in the input are extracted. These invariant features are then passed to the next layer. The features in the next layer are convoluted with different filters to extract more abstract features. In this way the convolutional layers are sensitive to features without position reference. The weights are adjusted during training of the CNN. The set of weights for the specific CNN at a certain training stage forms a model with the weights being its parameters. Additionally, some parameters are adjusted beyond the training process. These are called hyper-parameters. Examples of hyper-parameters are the number and size of the filters in the layers and the learning rate.

The CNN algorithm is a supervised deep learning algorithm. It infers a function that maps an input and output from the training data, which consists of a set of training examples with annotations. They represent a desired output that is referred to as a ground truth. These annotations are class labels in the case of classification. For the object detectors they are supplemented by the position of the object [33, 34], typically represented by a bounding box surrounding the object. The neural networks are trained using a gradient descent optimization algorithm [35]. The optimization algorithm minimizes the so-called loss function by updating the parameters of the model. The loss function is a specially designed function that serves as a metric of an error between predictions of the model and ground truth. The parameters are updated according to the chain rules [36]. The learning rate is the hyper-parameter that controls how much the parameters of the model are changed in response to the model error. CNNs usually use a stochastic gradient descent (SGD) algorithm [37, 38] for optimizing the loss function during training. The weights are updated based on random subsets (batches) of the training data rather than the complete training set. The period during training when the network has seen one batch is called iteration.

The YOLOv2 has a plain architecture with a moderate depth (see figure 2(a)). The base of YOLOv2 is the Darknet-19 classification network consisting of 19 convolutional layers. It was pre-trained on 1000 classes of images from ImageNet [39]. The YOLOv2 program accepts images of any size, converts them to the size of 416×416 pixels, and uses them as inputs for the CNN. The YOLOv3 software (see figure 2(b)), in contrast to YOLOv2, has a residual network architecture [40] and is based on the Darknet-53 classification network with 53 convolutional layers. The residual blocks are necessary to avoid possible performance degradation with increasing depth of the network. Darknet-53 was pre-trained on the same 1000 classes of the data set at ImageNet. YOLOv3 converts input images to the size of 608×608 pixels and uses them as an input for the CNN.

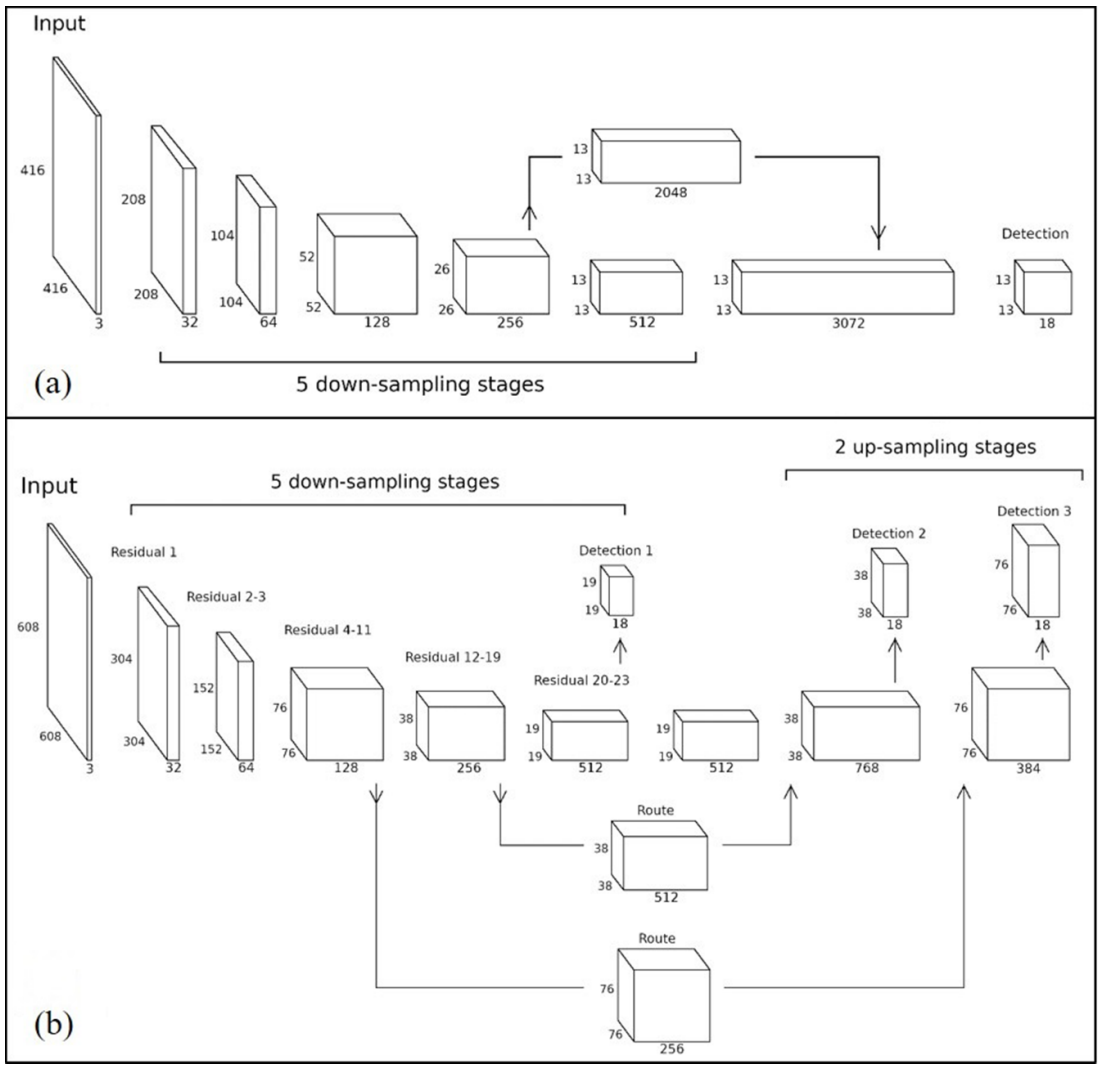

Figure 2. Architecture of CNN realized in YOLOv2 (a) and YOLOv3 (b). Both networks accept images with three color layers and a fixed size of 416 × 416 pixels (YOLOv2) and 608 × 608 pixels (YOLOv3). The data undergo five down-sampling stages to limit the number of parameters. The dimensions of the input for every stage are shown in the figure. In the case of YOLOv2 the data flow consequently from one layer to another. In the case of YOLOv3 five down-sampling stages are followed by two up-sampling stages. Every up-sampling stage receives additional activation from the corresponding down-sampling stage (marked 'Route'). Every down-sampling stage contains one to several residual blocks (marked 'Residual').

The loss functions of YOLOv2 and YOLOv3 as object detectors have terms associated with classification and localization (see [29] and [31] for details). We have modified the detection part of the networks to perform detection of objects of one class (single hits). An image was classified as a single hit when the network detected a single hit in the image. In the opposite case, when the network did not detect a single hit in the image, we classified it as not a single hit (binary classification). Default detection thresholds of 0.24 and 0.5 were applied for the networks YOLOv2 and YOLOv3, respectively.

2.4. Metrics

The main metrics to evaluate the classification results in the case of binary classification are accuracy, precision, and recall. Accuracy is the ratio of correct predictions among the total number of predictions made. Accuracy alone may not be informative if the frequency of single hits is a small fraction of the whole data set and it is therefore also useful to know the precision and recall. Precision is a measure of classifier exactness. Recall is a measure of classifier completeness. These metrics are defined as

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \tag{1}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{2}$$

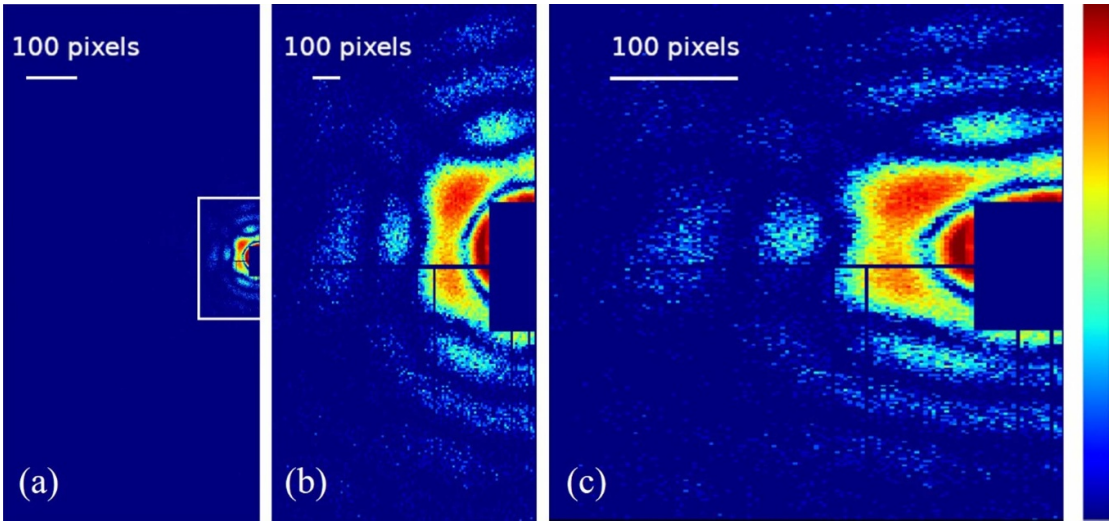

Figure 3. (a) Half of the detector panel used in the experiment; the white rectangle around the center of the diffraction pattern shows the area that was used for the conversion; (b) central part of the diffraction pattern converted into a color image and used for the analysis; (c) image modified by YOLOv2 ready to be used as an input to the neural network. Intensity in all three images is shown on a logarithmic scale.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{3}$$

where TP and TN are the number of true positive and true negative results and FP and FN are the number of false positive and false negative results, respectively.

The F-score metrics convey the balance between precision and recall. If the same importance is given to precision and recall, i.e. false positives are as undesirable as false negatives, the F_1-score can be used:

$$\text{F}_1 = 2\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{4}$$

The metric to compare the different outcomes of classification is the intersection over union (IoU). In our case it is defined as

$$\text{IoU} = \frac{\text{Intersection}}{\text{Union}} = \frac{\text{Intersection}}{S_1 + S_2 - \text{Intersection}}, \tag{5}$$

where Intersection is the number of patterns classified as single hits that are common for the given CNN model and a reference method, S_1 is the number of patterns classified as single hits by the CNN model, and S_2 is the number of patterns classified as single hits by the manual selection method.

2.5. Image processing

Below we describe an image processing step for conversion of the experimental data to be used as input to the CNN. The central part of each diffraction pattern with a size of 123 × 240 detector pixels containing the major part of the photon counts was selected. Then, it was up-sampled to the size of 954 × 1855 pixels and saved as an image using either a color or grayscale scheme, and either a linear or logarithmic scale. This conversion scheme with the initial up-sampling ensures that each detector pixel is represented as a monochrome rectangle in the image used as an input to the neural network. A trained CNN extracts features like edges and corners from the sharp borders between the monochrome rectangles representing the detector pixels. The starting and processed diffraction patterns to be used as input for YOLOv2 are shown in figure 3. For the color representation, photon counts either on a linear or logarithmic scale were converted into red-green-blue (RGB) layer images according to the 'jet' color map. For the grayscale representation, photon counts on linear and logarithmic scales were converted into one grayscale layer. In our implementation the three identical grayscale layers were stacked into a three-layer image. With this conversion all data were normalized to the maximum value of intensity of each diffraction pattern. Three color channels for linear and logarithmic representations of color images used as an input to the CNN are presented in figure 4.

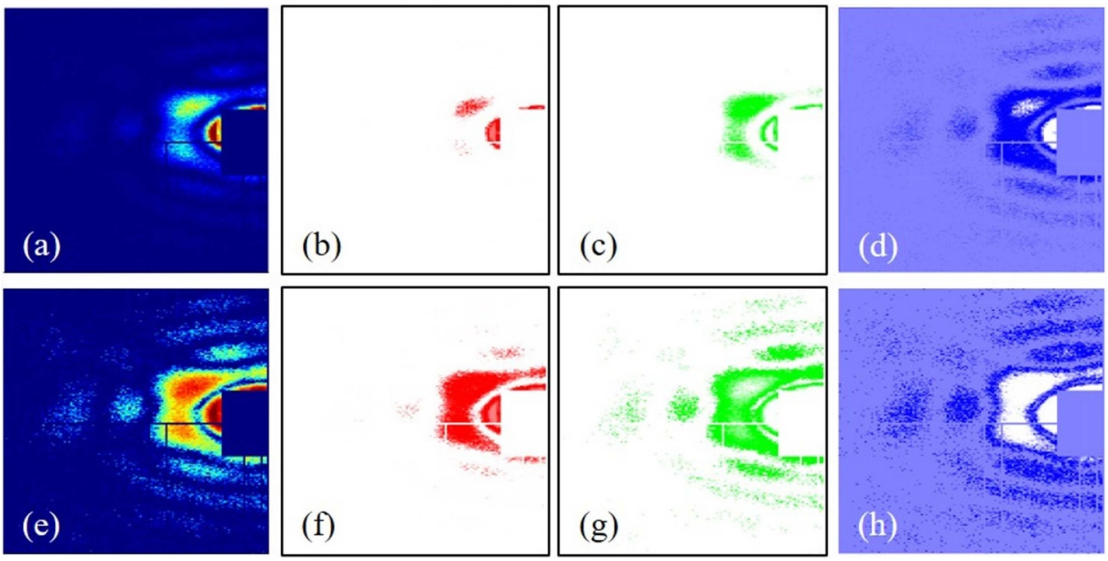

Figure 4. Constituent color layers for the two examples of the color images used for the CNN input linear (a) and logarithmic (e) scale (red (b), (f), green (c), (g) and blue (d), (h)).

Table 1. Number of diffraction patterns in different data sets.

Data set	Number of diffraction patterns (positive/negative)
Particle size 55–84 nm	18 213 (1196/17 017)
Training set	555 (165/390)
Validation set	336 (53/283)
Test set	17 932 (995/16 937)

2.6. Training, validation and test

Building a data set to train the model is a crucial step. A good training set should be representative and balanced. That means that the training set should represent the data that the model attempts to describe as well as possible. The total number of training examples should be sufficiently large to train the CNN for the full depth. Our aim is to use transfer learning with a limited amount of training data. To achieve this goal, we utilized pre-trained weights, i.e. weights that were obtained by training with a large data set (ImageNet in our case) as a starting point for our specific training set with a limited amount of data. As a result of CNN training the weights in the last layers are most affected and adjusted to the selected small training data set.

The ratio of the number of examples for each class should be close to what is expected in the experimental data set. However, it is difficult to achieve this goal using a small training data set. In our case, based on our previous experience, we expect that the number of positive examples (single hit) will be much lower than the number of negative ones (no single hit). If we consider a similar ratio of positive and negative examples in the training set we could have a situation in which negative examples are selected with high probability during each iteration. As a compromise, our training set contained 165 positive and 390 negative examples (see table 1).

A validation set is used to evaluate a given model, and the same requirements as for the training set apply to it as well. We kept the total number of examples small, although the ratio of positive to negative examples could be lower in the case of validation. The validation set in our case contained 53 positive and 283 negative examples (see table 1).

From the initial data set containing 18 213 patterns, which were selected in the particle size range from 55 nm to 84 nm, a test data set of 17 932 patterns was determined (see table 1). It was obtained from the initial data set by removing 281 patterns belonging to the same particle size range from it that were used for training or validation. In a similar way, by removing patterns of the training or validation set from the manually selected 1196 single hits, we finally obtained 995 single hit patterns that were used as a ground truth for testing the performance of the CNN.

The images for color (linear and logarithmic scale) and grayscale (linear and logarithmic scale) representations showing positive and negative examples of our training set are displayed in figure 5. All training data had ground truth annotations. Annotations for the positive examples contain class labels (single hit) and the coordinates of the bounding box covering the major part of all intensity counts in the

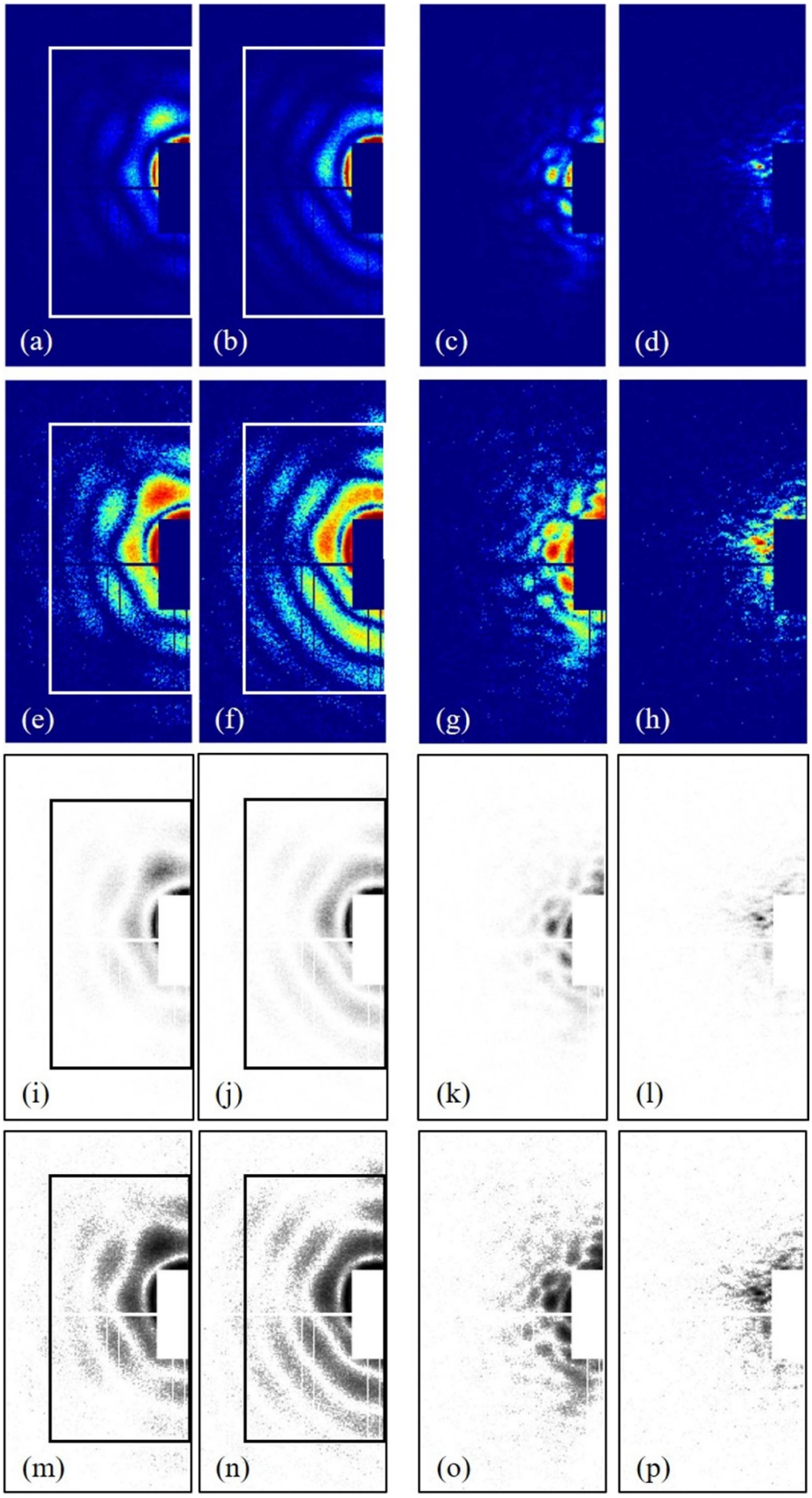

Figure 5. Different representations of the training data: three RGB layers, 'jet' color scheme, linear scale to photon counts (a)–(d); three RGB layers, 'jet' color scheme, logarithmic scale to photon counts (e)–(h); three identical grayscale layers, linear scale to photon counts (i)–(l); three identical grayscale layers, logarithmic scale to photon counts (m)–(p) positive examples (single hits)—(a), (b), (e), (f), (i), (j), (m) and (n); negative examples—(c), (d), (g), (h), (k), (l), (o) and (p). Corresponding bounding box annotation is shown for positive examples.

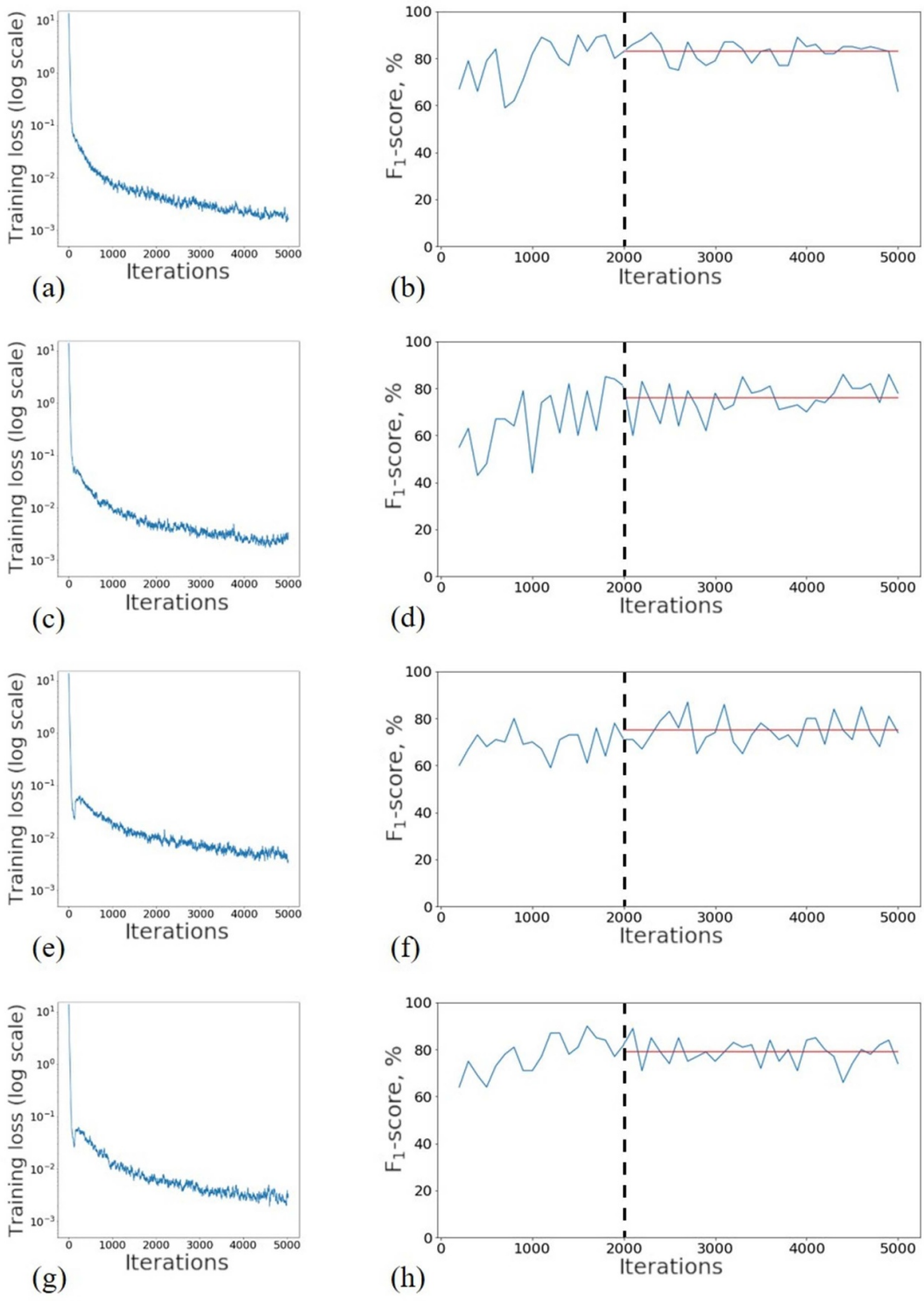

Figure 6. (a), (c), (e), (g) Value of the training loss function as a function of iterations; (b), (d), (f), (h) F_1-score calculated for validation data set as a function of iterations. Saturation of the F_1-score is reached after 2000 iterations (shown by black vertical dashed lines). Red lines show the mean value of the F_1-score at saturation level. Different model families of YOLOv2 are used here: (a), (b) color image, linear scale; (c), (d) color image, logarithmic scale; (e), (f) grayscale image, linear scale; (g), (h) grayscale image, logarithmic scale.

image. To accelerate the annotation process, we considered the same coordinates of the bounding box for all positive examples in the training set of the data as shown in figures 5(a), (b), (e), (f), (i), (j), (m), and (n). The validation and test data sets had annotations in the form of class labels only (single hit or no single hit) as we were interested in correct classification but not in localization.

During training the value of the loss function (training loss) was calculated after each iteration (see figures 6(a), (c), (e), (g) and 7(a), (c)). One can see that the training loss was gradually decreasing as a

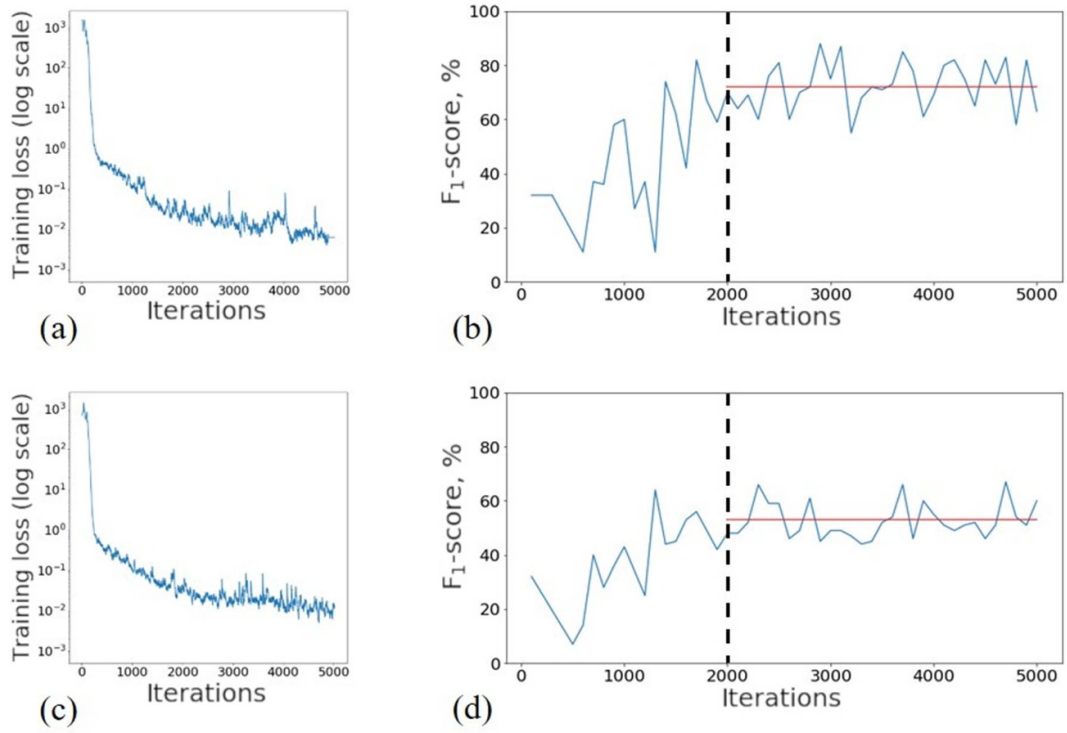

Figure 7. As in figure 6 for the different model families of YOLOv3: (a), (b) color image, linear scale; (c), (d) grayscale image, linear scale.

function of iterations, which was a good sign of CNN convergence. The local increase in the training loss in some plots corresponds to the point where the training rate was changed. The set of weights for the CNN was saved after 100 iterations, giving a model as described above. A set of models at different training stages for a certain network architecture (YOLOv2 or YOLOv3) and data representation (color or grayscale images, linear or logarithmic scale) constitutes a family of models.

We used validation exclusively to find the point to stop training. The YOLO default hyper-parameters were considered as being optimal and were not changed during validation. We made a binary classification on the validation set applying the YOLO software with the weights obtained during the training process, which was performed for each family of models. The results of classification at the validation step were compared to the ground truth and the F_1-score was calculated (see figures 6(b), (d), (f), (h) and 7(b), (d)). The F_1-score reached saturation after about 2000 iterations for every family of models. This means that every model in the family may be regarded as optimal after 2000 iterations. The F_1-score did not decrease within 5000 iterations, indicating that no effect of overfitting is present. Table 2 summarizes the results of the mean values and population standard deviation of accuracy, precision, recall, and F_1-score calculated for the validation set. As follows from this table, the model families have high values of accuracy and precision, but significantly lower values of recall. This behavior is because in the validation set we have a large number of negative examples that are correctly identified. This means that in this case we have a comparably large number of TN that determines high values of accuracy. At the same time, we have a small number of FP cases that effectively makes precision values high as well. In the case of recall, its value is determined by comparably large values of FN, which are of the same order as TP. This is because CNN does not recognize all diffraction patterns labeled as positive in the validation set.

3. Results

We used an NVIDIA GeForce GTX 1060 with 6 GB of internal memory for the training, validation and final tests using our CNN. The training time for 100 iterations was on average about 280 s for YOLOv2 and 290 s for YOLOv3. This training time was almost the same for both networks due to the smaller batch size for the deeper network YOLOv3. The processing time for one image was about 20 ms for YOLOv2 and about 75 ms for YOLOv3. The difference in processing time between YOLOv2 and YOLOv3 was due to the difference in image size and depth of the networks.

Table 2. Mean and population standard deviation (std) values (from 2000 to 5000 iterations) of the accuracy, precision, recall and F_1-score for the different families of YOLOv2 and YOLOv3 models calculated with respect to the validation data set.

Family of models	Acc. mean, %	Acc. std, %	Prec. mean, %	Prec. std, %	Recall, mean, %	Recall, std, %	F_1-score mean, %	F_1-score std, %
YOLOv2, color, linear	95	1	98	2	72	8	83	5
YOLOv2, color, log	94	1	92	4	64	9	76	7
YOLOv2, grayscale, linear	93	1	91	4	64	9	75	6
YOLOv2, grayscale, log	94	1	98	3	66	7	79	5
YOLOv3, color, linear	93	2	94	5	60	13	72	9
YOLOv3, grayscale, linear	89	1	83	7	39	8	53	6

Table 3. Number of single hits classified by the models of different model families, population standard deviation (std), and number of single hits in the stable selection.

Training stage model	N of iterations 4000	N of iterations 4100	N of iterations 4200	N of iterations 4300	N of iterations 4400	Population std	Number of single hits
YOLOv2 color	1247	1376	1632	1604	1509	144.3	1185
YOLOv2 color log	1664	1633	1414	1793	2060	212.1	1368
YOLOv2 grayscale	2990	3046	2691	3847	2194	539.3	1756
YOLOv2 grayscale log	1506	2255	1605	1297	926	435.9	904
YOLOv3 color	1063	3497	4566	3036	1958	1215.2	1041
YOLOv3 grayscale	3452	3377	3135	2541	5014	820.1	2316

We performed classification on the test data set. We considered 4000 iterations as optimal as soon as it was far from the saturation value of 2000 iterations and within 5000 iterations no effect of overfitting was observed. As soon as the F_1-score showed comparably large population standard deviation we considered five consecutive models for each family (differing by 100 iterations starting from the 4000th iteration). The number of single hits selected by each of these models for each network architecture and different data representation is given in table 3. First of all, we observed how stable each model was. The population standard deviation in the number of selected single hits at different training stages for a given model was taken as a measure of stability. These stages are considered to have almost the same and optimal level of training and should give nearly the same number of single hits, if the model is stable. As seen in table 3, the spread in the number of selected single hits among the models of the same architecture (YOLOv2 or YOLOv3) is due to the relatively small size of the training set, arbitrary orientation, and finite number of positive examples in the training set. The YOLOv2 models have significantly narrower distribution in the number of selected single hits in contrast to the YOLOv3 models. The lower stability for the YOLOv3 models can be explained by the smaller batch size and larger number of parameters due to the higher depth of the network.

In order to mitigate the instabilities in single hit selection, we applied the following strategy. The diffraction pattern was classified as a single hit only if it was classified as a single hit in each of the five consequent models (see table 3). We see from this table that the lowest standard deviation is observed for the YOLOv2 color images case. In fact, this has happened occasionally due to a small distribution of F_1-score values in this region of iterations (see figure 6(b)).

In order to compare the performances of different networks, we evaluated the intersection and IoU of our results with the manual selection set of data, calculated accuracy, precision and recall. Inspection of table 4 shows that the performances of both networks for color images on a linear scale are similar. The performance of the YOLOv2 model trained on grayscale images on a linear scale is significantly better than that of YOLOv3. This serves as an indication that a relatively shallow neural network is sufficient for single hit classification. The models trained on grayscale images perform worse than those trained on color images, which is more pronounced in the case of YOLOv3. Features extracted from three different color layers seem to have more information than those extracted from the one grayscale layer. We expected that the model with a logarithmic scale of intensity should perform significantly better than the linear-scale one as soon as the scattering signal decays as $I(q) \approx q^{-3} \div q^{-4}$ [41]. Table 4 shows that both models perform rather similarly.

Table 4. Comparison of the stable selection for different CNN architectures, and data representation with the manual selection, which is considered as a ground truth. This manual selection is chosen to estimate the intersection, IoU, accuracy, precision, and recall.

Model	Number of single hits	Intersection with the manual selection	IoU for manual selection (%)	Accuracy (%)	Precision (%)	Recall (%)
YOLOv2 color, linear	1185	597	38	95	50	60
YOLOv2 color log	1368	614	35	94	45	62
YOLOv2 gray-scale, linear	1756	622	29	92	35	63
YOLOv2 gray-scale log	904	487	34	95	54	50
YOLOv3 color, linear	1041	505	33	94	49	51
YOLOv3 gray-scale, linear	2316	465	16	87	20	47

We propose that this could be because the features extracted by the CNN from color representation of diffraction patterns on linear or logarithmic scales are similar. This may be valid for the range of low total intensities that we have in the considered SPI experiment. We also note (see table 4) that the grayscale logarithmic images perform similarly to the color images; however, the performance of the grayscale linear images is worse. We infer that the feature extraction from the grayscale linear images is less informative in comparison to color images.

It should be mentioned here that there is a certain discrepancy between the values of precision calculated on the validation and test data sets (compare tables 2 and 4). Much higher values of precision during validation mean much smaller values of FP compared to TP. This could be an indication that annotations for the validation data set were more consistent than those for the test data set.

If we consider the manual selection as a ground truth, the accuracy for the YOLOv2 color linear model calculated for the test set of 17 932 patterns is around 95% with the precision and recall being 50% and 60%, respectively. This result may be explained by the following considerations. The large value of accuracy is due to a large number of TN values. In this case, due to equation (1), accuracy may be close to 100%. This also indicates that accuracy alone is not a sufficient metric to describe the results of CNN classification. At the same time, the precision becomes much lower than in the case of the validation set due to a high value of FP (that is of the same order as TP) determined by the CNN, though visually part of them should belong to positive values. We note also that the recall values are similar to the ones determined for the validation set of data.

We calculated the scores for the EM-based selection for the data set of 18 213 patterns filtered by particle size. The IoU of the EM-based classification with the manual selection was 34% with accuracy of 94%, precision of 53% and recall of 48%. The scores are similar to those calculated for the YOLOv2 color linear model.

4. Conclusions and outlook

As we showed, classification of single hits in SPI experiments can be effectively performed by a CNN. We demonstrated that it is possible to extract single hits with a high level of accuracy with respect to the manual selection. A moderate depth of CNN, like that of YOLOv2, was sufficient for this task and the use of the deeper network YOLOv3 did not improve the efficiency. The spatial integrity of diffraction patterns does not seem to be crucial to the work of the CNN. We found similar performances for the classification utilizing the proposed CNN and the EM-based method. We found also that the efficiency of the CNN on the data set that was used for our tests does not depend on intensity expressed on a linear or logarithmic scale for color images. However, the performance of the CNN was worse in the case of grayscale images. This question will need more detailed studies in the future.

The results obtained here as a first attempt of using artificial intelligence methods for classification tasks in SPI experiments may be improved in the future by (a) making annotations more consistent, (b) optimizing the architecture of the network and its hyper-parameters, and (c) training the entire network on the large amount of corresponding data. In addition, transfer learning may be used more efficiently with further fine-tuning of the network trained on the corresponding data. We used transfer learning to train the

network for classification with a limited amount of training data taken from the experiment. Such a small training set can produce an additional uncertainty in the selection of single hits (as seen in table 3). To overcome this problem, instead of SGD, a different variant of the gradient descent method may be used. As an alternative approach, training of the CNN on a large set of simulated data may be used.

Using a CNN with transfer learning, in our opinion, is crucial as this enables automatization of the analysis pipeline. A proposed workflow foresees an expert identifying a training set as soon as the experiment starts. Then, the network can run for the rest of the experiment. Thus, one of the most challenging throughput problems of today's experiments, i.e. data reduction, can be addressed. Indeed, one could even foresee that the network decides which data are stored and which are not saved at all.

We examine a possible generalization of the CNN model for heterogeneous systems, i.e. systems composed by different conformations of a given particle, by implementing a convolutional auto-encoder (CAE) [42]. The CAE consists of two parts, named the encoder and decoder, trained together. The encoder attempts to learn a low-dimensional representation of the input data, while the decoder reconstructs the data using its low-dimensional representation. At the stage of training, when the input patterns coincide with the output patterns, we may assume that the network is fully trained and this set of data may be used as a ground truth. The CAE can be trained from input to output using a large amount of simulated data for different conformations of particles used in the SPI experiments. When the training is finished, the encoder part trained to produce a low-dimensional representation of input data can be used as a base for the classification network.

The proposed approach based on CNN for classification of large amount of data may be beneficial for applications at high repetition rate XFELs [5] while collecting data with the megahertz rate [43].

Data availability statement

The data that support the findings of this study are openly available at the following URL: https://cxidb.org/id-156.html

Acknowledgments

We acknowledge the support of the project and discussions with E Weckert, X Yang is acknowledged for his careful reading of the manuscript, and A Aquila for fruitful discussions in the frame of this project. The Helmholtz Association Innovation Pool Project AMALEA is acknowledged.

Funding

This work was supported by the Helmholtz Associations Initiative and Networking Fund (Grant No. HRSF-0002) and Russian Science Foundation (Grant No. 18-41-06001).

ORCID iD

Ivan A Vartanyants https://orcid.org/0000-0002-0340-8234

References

[1] Neutze R, Wouts R, van der Spoel D, Weckert E and Hajdu J 2000 Potential for biomolecular imaging with femtosecond x-ray pulses *Nature* **406** 752
[2] Gaffney J K and Chapman H N 2007 Imaging atomic structure and dynamics with ultrafast x-ray scattering *Science* **316** 1444
[3] Emma P *et al* 2020 First lasing and operation of an ångstrom-wavelength free-electron laser *Nat. Photonics* **4** 641
[4] Ishikawa T *et al* 2012 A compact x-ray free-electron laser emitting in the sub-ångström region *Nat. Photonics* **6** 540
[5] Decking W *et al* 2020 A MHz-repetition-rate hard x-ray free-electron laser driven by a superconducting linear accelerator *Nat. Photonics* **14** 391
[6] Chapman H N *et al* 2006 Femtosecond diffractive imaging with a soft-x-ray free-electron laser *Nat. Phys.* **2** 839
[7] Seibert M M *et al* 2011 Single mimivirus particles intercepted and imaged with an x-ray laser *Nature* **470** 78
[8] Hantke M F *et al* 2014 High-throughput imaging of heterogeneous cell organelles with an x-ray laser *Nat. Photonics* **8** 943
[9] van der Schot G *et al* 2015 Imaging single cells in a beam of live cyanobacteria with an x-ray laser *Nat. Commun.* **6** 5704
[10] Ekeberg T *et al* 2015 Three-dimensional reconstruction of the giant mimivirus particle with an x-ray free-electron laser *Phys. Rev. Lett.* **114** 098102
[11] Rose M *et al* 2018 Single-particle imaging without symmetry constraints at an x-ray free-electron laser *IUCrJ* **5** 727
[12] Assalauova D *et al* 2020 An advanced workflow for single particle imaging with the limited data at an x-ray free-electron laser *IUCrJ* **7** 1102
[13] Bobkov S A, Teslyuk A B, Kurta R P, Gorobtsov O Y, Yefanov O M, Ilyin V A, Senin R A and Vartanyants I A 2015 Sorting algorithms for single-particle imaging experiments at x-ray free-electron lasers *J. Synchrotron Radiat.* **22** 1345

[14] Dempster A P, Laird N M and Rubin D B 1977 Maximum likelihood from incomplete data via the EM algorithm *J. R. Stat. Soc. SeR.* B **39** 1 (https://www.jstor.org/stable/i349763)
[15] Krizhevsky A, Sutskever I and Hinton G E 2012 ImageNet classification with deep convolutional neural networks *Proc. Conf. Advances in Neural Information Processing Systems (NIPS)* p 25
[16] Szegedy C, Toshev A and Erhan D 2013 Deep neural networks for object detection *Proc. Conf. Advances in Neural Information Processing Systems (NIPS)* p 2553
[17] Shi Y *et al* 2019 Evaluation of the performance of classification algorithms for XFEL single-particle imaging data *IUCrJ* **6** 331
[18] Zimmerman J *et al* 2019 Deep neural networks for classifying complex features in diffraction images *Phys. Rev.* E **99** 063309
[19] Yang X *et al* 2020 Tomographic reconstruction with a generative adversarial network *J. Synchrotron Radiat.* **27** 486
[20] Ferguson K R *et al* 2015 The atomic, molecular and optical science instrument at the Linac Coherent Light Source *J. Synchrotron Radiat.* **22** 492
[21] Osipov T *et al* 2018 The LAMP instrument at the Linac Coherent Light Source free-electron laser *Rev. Sci. Instrum.* **89** 035112
[22] Aquila A *et al* 2015 The linac coherent light source single particle imaging road map *Struct. Dyn.* **2** 041701
[23] Li H *et al* 2020 Diffraction data from aerosolized coliphage PR772 virus particles imaged with the Linac Coherent Light Source *Sci. Data* **7** 404
[24] Reddy H K N *et al* 2017 Coherent soft x-ray diffraction imaging of coliphage PR772 at the Linac Coherent Light Source *Sci. Data* **4** 170079
[25] Nazari R *et al* 2020 3D printing of gas-dynamic virtual nozzles and optical characterization of high-speed microjets *Opt. Express* **28** 21749
[26] Benner W H, Bogan M J, Rohner U, Boutet S, Woods B and Frank M 2008 Non-destructive characterization and alignment of aerodynamically focused particle beams using single particle charge detection *J. Aerosol Sci.* **39** 917
[27] Strüder L *et al* 2010 Large-format, high-speed, x-ray pnCCDs combined with electron and ion imaging spectrometers in a multipurpose chamber for experiments at 4th generation light sources *Nucl. Instrum. Methods Phys. Res.* A **614** 483
[28] Damiani D *et al* 2016 Linac Coherent Light Source data analysis using psana *J. Appl. Crystallogr.* **49** 672
[29] Redmon J and Farhadi A 2016 YOLO9000: better, faster, stronger (arXiv:1612.08242)
[30] Redmon J and Farhadi A Darknet: open source neural networks in C (available at: https://pjreddie.com/darknet/)
[31] Redmon J and Farhadi A 2018 YOLOv3: an incremental improvement (arXiv:1804.02767)
[32] Goodfellow I, Bengio Y and Courville A 2016 *Deep Learning* (Cambridge, MA: MIT Press)
[33] Everingham M 2010 The Pascal visual object classes (VOC) challenge *Int. J. Comput. Vis.* **88** 303
[34] Russakovsky O *et al* 2015 ImageNet large scale visual recognition challenge (arXiv:1409.0575)
[35] Ruder S 2017 An overview of multi-task learning in deep neural networks (arXiv:1706.05098)
[36] Kaplan W 1984 *Advanced Calculus* 3rd edn (Reading, MA: Addison-Wesley)
[37] Bottou L, Curtis F E and Nocedal J 2017 Layer normalization (arXiv:1607.06450)
[38] Goyal P *et al* 2017 Accurate, large minibatch SGD: training ImageNet in 1 hour (arXiv:1706.02677)
[39] Fei-Fei Li *et al* ImageNet (available at: www.image-net.org)
[40] He K *et al* 2015 Identity mappings in deep residual networks (arXiv:1603.05027)
[41] Rose M *et al* 2018 Quantitative ptychographic bio-imaging in the water window *Opt. Express* **26** 1237
[42] Masci J *et al* 2011 Stacked convolutional auto-encoders for hierarchical feature extraction *Artificial Neural Networks and Machine Learning—ICANN 2011* (*Lecture Notes in Computer Science*) eds T Honkela, W Duch, M Girolami and S Kaski (Berlin: Springer) vol 6791
[43] Sobolev E *et al* 2020 Megahertz single-particle imaging at the European XFEL *Commun. Phys.* **3** 97