# Deep Learning for Bragg Coherent Diffraction Imaging: Detector Gap Inpainting and Phase Retrieval

# Thesis

présentée et soutenue publiquement le

Pour l'obtention du titre de

Docteur de l'Université Grenoble Alpes
(mention Physique du rayonnement et de la matière condensée)

par
Matteo Masto

sous la direction de
Dr. Tobias Schülli, Dr. Vincent Favre-Nicolin, Dr. Steven Leake

**Composition du Jury**

| | | |
|---|---|---|
| XXXXXXXXXX | PR, XXXXXXXXXX | **Rapporteur** |
| XXXXXXXXXX | PR, XXXXXXXXXX | **Rapporteur** |
| XXXXXXXXXX | PR, XXXXXXXXXX | **Examinateur** |
| XXXXXXXXXX | PR, XXXXXXXXXX | **Examinateur** |
| Tobias Schülli | ESRF | **Directeur de Thèse** |
| Vincent Favre-Nicolin | ESRF UGA, | **Directeur de Thèse** |
| Steven Leake | ESRF | **Directeur de Thèse** |

**École Doctorale de Physique-Grenoble**

# Contents

Chapter 1

# Introduction

## 1.1 Introduction

The present document is a draft of my PhD manuscript.

# Chapter 2

# Bragg Coherent Diffraction Imaging

## 2.1 Single crystal diffraction

## 2.2 Phase Problem

Chapter 3

# Convolutional Neural Networks

## 3.1 Introduction on neural networks

## 3.2 Convolutional

## 3.3 U-Net and MSD-Net

# Chapter 4

# Deep Learning for Detector Gaps Inpainting

In this chapter the "detectors' gaps problem" in Bragg Coherent Diffraction Imaging and our approach to solve it using Deep Learning are discussed. The main state-of-the-art measures are presented briefly and the topic of image inpainting with Deep Learning is introduced. The focus will then shift to our works that led eventually to the optimal "Patching-based" approach that can also be found in the published paper entitled *"Patching-based deep learning model for the Inpainting of Bragg Coherent Diffraction patterns affected by detectors' gaps"* (https://doi.org/10.1107/S1600576724004163). The chapter is closed with some analyses of the performances of the DL models in a variety of simulated and experimental cases.

## 4.1 The "Gap Problem"

At time of writing, standard BCDI experiments employ pixelated photon counting detectors to acquire the diffraction patterns. These detectors can guarantee high spatial resolution, noise-free counting and fast read-out times. Two examples of these devices, currently used at the ID01 beamline are the MAXIPIX and EIGER detectors [1, 2]. These detectors are often built by tiling together several sensing chips in order to cover a larger area, and are typically bonded to an Application-Specific Integrated Circuit (ASIC) using bump bonding. This implies the presence, in the overall sensing region, of vertical and/or horizontal stripes that are not sensitive to the impinging radiation. The width of these lines varies depending on the device but normally does not exceed the equivalent of some tens of pixels. Specifically, for the MAXIPIX detector the gap size is 6 pixels wide while the EIGER has two types of larger gaps of 12 pixels and 38 pixels width. The detector gaps problem does not affect BCDI only, but it is shared among other x-ray techniques that deal with single photon-counting pixelated detectors and/or beamstops. We have seen in chapter 1.1 that during a BCDI scan the 2D images acquired by the detector are stacked to form a 3D array. This leads these lines to become planes of missing signal in the dataset. The problems arise when reconstructing the data affected by these gaps. In fact, these regions of non-physical zero intensity deceive the Phase Retrieval algorithms inducing the presence of artifacts in the reconstructions[3].

It follows that the reliability of the reconstructions in this case is compromised as the strain

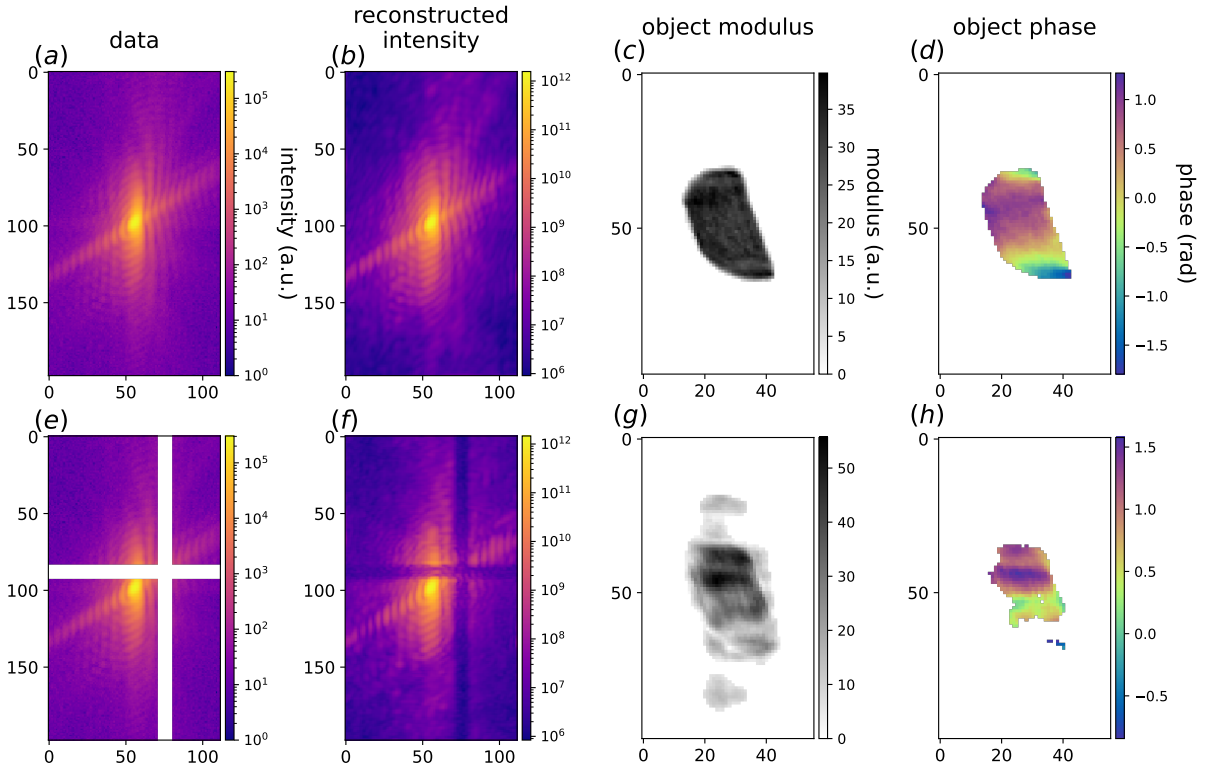


**Figure 4.1: Effect of detector gaps in BCDI reconstructions (a)** The central xz slice of an experimental diffraction pattern. **(b)** The same slice of the diffracted intensity calculated from the retrieved object. **(c - d)** xz slice of the modulus and phase respectively of the particle obtained from the phasing of the gap-less dataset. **(e)** Same slice as in **(a)** with an artificially added 9 pixel-wide, cross-shaped gaps to mimic the detector's ones. **(f)** The same slice of the diffracted intensity calculated from the retrieved object when not masking the gap regions. **(h - g)** xz slice of the modulus and phase respectively of the particle obtained from the phasing of the gap-affected dataset. The distortions caused by the gaps are evident.

distribution can be deeply affected by the artifacts. A good practice during standard BCDI experiments is to avoid the gaps by moving the detector if possible. However, this tends to be problematic for the case of high-resolution BCDI, i.e. when the diffraction pattern measurement extends to higher q-values, thus covering more than one sensing chip and necessarily crossing a gap region. Under these circumstances it becomes important to reduce the amount of artifacts deriving from the gaps.

## 4.2 State of the art

Here we will discuss the current strategies employed to treat the detector gaps. As someone could argue, the simplest yet not practical, solution would be to slightly move the detector sideways and acquire a second full scan with the gap hiding a different region of the same Bragg peak, and then merge the two measurements into a single gap-less one. This would more than double the acquisition time making it, de facto, never an option during standard experiments.

The PyNX software, routinely used for the BCDI phase retrieval at ID01, allows the user to

define a mask of the gap regions and ignore those pixels during the execution. In this way the quality of the reconstruction improves, but one can still notice the presence of high-frequency oscillations appearing in both object's modulus and phase. The origin of these artifacts can be found in the diffracted intensity calculated from the reconstructed particle as one can clearly see that the gap-regions is filled with nonphysically high intensity (see Fig. 4.2)

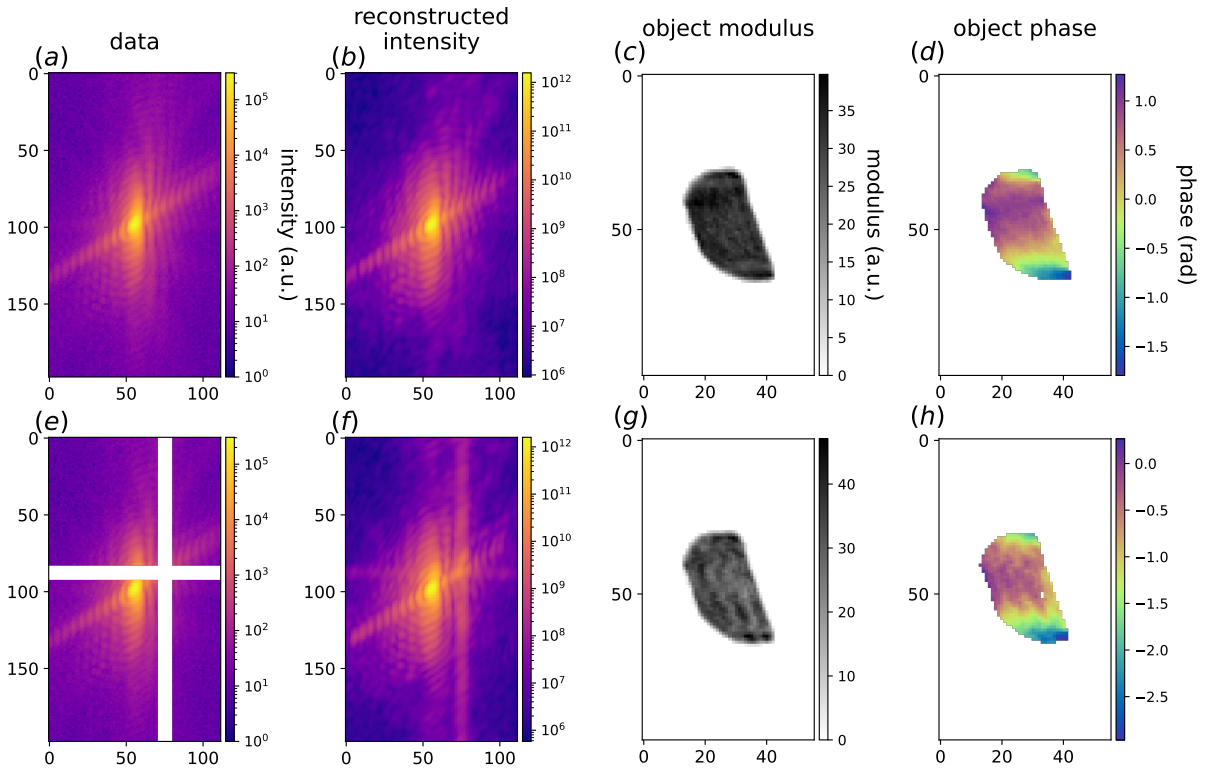


**Figure 4.2: Masking the gap region during phasing (a)** The central xz slice of an experimental diffraction pattern. **(b)** The same slice of the diffracted intensity calculated from the retrieved object. Comparing this figure with 4.1**(b)** one can see that when excluding the gap region from the phasing with a mask, the calculated intensity shows bright non-physical streaks instead of the gaps. **(c - d)** xz slice of the modulus and phase respectively of the particle obtained from the phasing of the gap affected data with a mask of the gap regions. Despite the much higher quality of the reconstruction, one can notice some oscillatory artifacts appearing in both the modulus and the phase of the retrieved object.

Another, more invasive, option is to *fill* these gaps with an estimate of the intensity distribution that would be there, before the phase retrieval. These tasks of filling gap in images is usually referred to as "inpainting". The following paragraph mentions the most relevant inpainting methods to give a context for our work.

### 4.2.1 Background on Image Inpainting Research

Computational image inpainting has been widely studied in the field of photography and imaging for many years [4, 5]. The inpainting problem can be defined as the task of utilizing known information extractable from the image, to repair the parts where this information is missing, where for known information the colors, the textures and the semantic features are intended. In the history of image inpainting a clear cut can be observed when deep learning

methods have started to be employed. For traditional inpainting, different techniques have been explored, from the texture synthesis methods pioneered by Efros and Leung [6] to the use of PDEs as Navier-Stokes equations proposed by Bertalmio *et al.* [7] and then again from sparse representations [8] to hybrid methods combining variational and statistical methods [9]

More recently instead, Deep Learning models, headed by Convolutional Neural Networks (CNN), have taken the place of more traditional methods as they can attain higher accuracy for more complex inpainting tasks. By undergoing a training process, CNNs can "learn" to recognize and reproduce the semantic features of the training dataset, and thus leverage them during inference as additional information beside the colors and textures of the specific image to restore. As we have seen in **??**, the typical CNN architecture for image generation consists of an encoder, which retains the features of the input image and compresses them into a lower dimensional latent space, and a decoder, which is responsible for the generation of the output image starting from the latent space. The model are then trained according to a loss function that pushes the model's predictions to be close to a given ground truth reference. In some cases, the loss function can be replaced by another CNN that is trained to discriminate true images from the ones predicted by the model. These complementary networks are known as Generative Adversarial Networks (GAN), firstly proposed by Goodfellow *et al.* [10], and have also been used for image inpainting (e.g. [11]). Since reviewing the vast amount of works about CNN for image inpainting is beyond the scope of this thesis and for more information, we redirect the reader to the reviews published by Elharrouss *et al.* and Xu *et al.* [5, 12]. as well as this blog article [13]. For what concerns the application of DL based inpainting for scientific imaging, early works date back to 2018 as in the case of Sogancioglu *et al.* for x-ray human chest 2D radiographic images [14] and to 2020 for 2D microscopic images [15]. A couple of years later Tanny Chavez and coauthors published a paper comparing the performances of different CNN models for the inpainting of 2D x-ray diffraction images [16]. The work is precisely addressing the gap problem for x-ray detectors used for powder diffraction measurements and is awarding UNet and Mixed Scale Dense (MSD) models for the best performances on experimental data. The DL models outperform interpolations obtained with biharmonic functions across 7 and 17 pixel-wide gaps. This work has been of inspiration for the design of our DL model for BCDI gaps inpainting. In the same year, another work on DL based inpainting for x-ray detector gaps was published by Alfredo Bellisario and coauthors [17]. The authors tested a UNet-like model on the inpainting of 2D simulated, noiseless coherent diffraction patterns against gaps of different sizes (2 to 20 pixels) along the central row. The gaps were placed such that the center of the peak was covered, a choice that, as we will see later, yields better results than predictions on peripheral areas. To our knowledge, at the time of writing, no other works about deep learning based inpainting for X-ray detector gaps are present in the literature.

## 4.3 Model design: 2D case

On the heels of the last mentioned works we have started to tackle the detector gaps problem for BCDI using CNNs. For simplicity, we started off with 2D case, using simulated diffraction patterns and inpainting randomly placed vertical gaps of different width. First, we created a training set of simulated data, composed of pairs of gap-affected images and corresponding gap-free ground truths, then built a UNet-like model and trained it in a supervised fashion.

### 4.3.1 Dataset creation

The creation of training datasets of simulated 2D BCDI patterns for both the gap-inpainting and phase retrieval tasks has followed the procedure described in this paragraph.
In first place, once chosen the size of the array, a randomly shaped polygon is created in the center using `scipy.spatial.ConvexHull` function. This guarantees the object to have a compact support with homogeneous electron density as assumed for BCDI. Subsequently, a random phase field of the same size with variable phase range and correlation length is generated thus the complete complex object is formed. In order to make the object more realistic a Gaussian filter and Gaussian random noise are applied to the object's modulus, so to smoothen the edges and simulate real cases respectively. At this point the object is resized to the shape required to match the chosen oversampling ratio and the 2D Discrete Fourier Transform is computed. As last stage, Poisson noise is added to the diffraction patterns with different magnitudes to simulate various X-ray flux conditions.
Datasets contain a number of diffraction patterns in the order of thousands and for each of them the random variables are different as well as the oversampling ratios. In the datasets for the training of phase retrieval models, the reciprocal space phase corresponding to each diffraction pattern is saved as well and used as ground truth label. For inpainting tasks a randomly located vertical gap mask was created and applied to the intensity data. In some cases cross-shaped gaps were added instead to simulate the experimental condition of the Bragg peak in the vicinity of the corner of the sensing area. The size of the gaps was chosen to be consistent across the dataset and four different cases were studied (3px, 6px, 9px, 12px).

### 4.3.2 2D Model design

The 2D model that we have implemented is a UNet that takes in input batches of 32 simulated BCDI patterns affected by both vertical and cross-shaped gaps. Each diffraction pattern is transformed into logarithmic scale to enhance the spatial features and then normalized between 0 and 1. This last passage is proven to be convenient to any DL model [18]. Regarding the logarithmic transformation, it is important to notice that in order to avoid problems for zero intensity values, the $\log(I+1)$ was taken. The shape of each image was chosen to be of $128 \times 128$ pixels. The inputs go through five convolutional blocks inside each of which a convolutional layer, a Leaky ReLU activation function and a MaxPooling operation are applied. The tensor's dimensionality is so reduced down to $2 \times 2$ while the channel dimension is brought up to 768 filters while the kernel size is kept at $3 \times 3$. In this first model we directly pass this $(32, 2, 2, 768)$ tensor to the decoder that, mirroring the encoder, is composed of five blocks inside each of which there is a convolutional layer, a Leaky ReLU activation function and a UpSampling layer. We also implemented skip connections connecting each encoder block to its corresponding shape-like decoder block. This measure has proven to be beneficial for the information flow between encoder-decoder [**skip**]. The last activation layer of the model is a sigmoid function that guarantees an output bounded between 0 and 1

In the first place we utilized a simple Mean Squared Error (MSE) as cost function inside the gap region only, training the model on 12'000 diffraction patterns over 10 epochs, with ADAM optimizer and a learning rate of $10^{-4}$. We have tested the Mean Absolute Error (MAE) and the Structural Similarity Index Measure (SSIM) [19] as well afterwards and compared the results

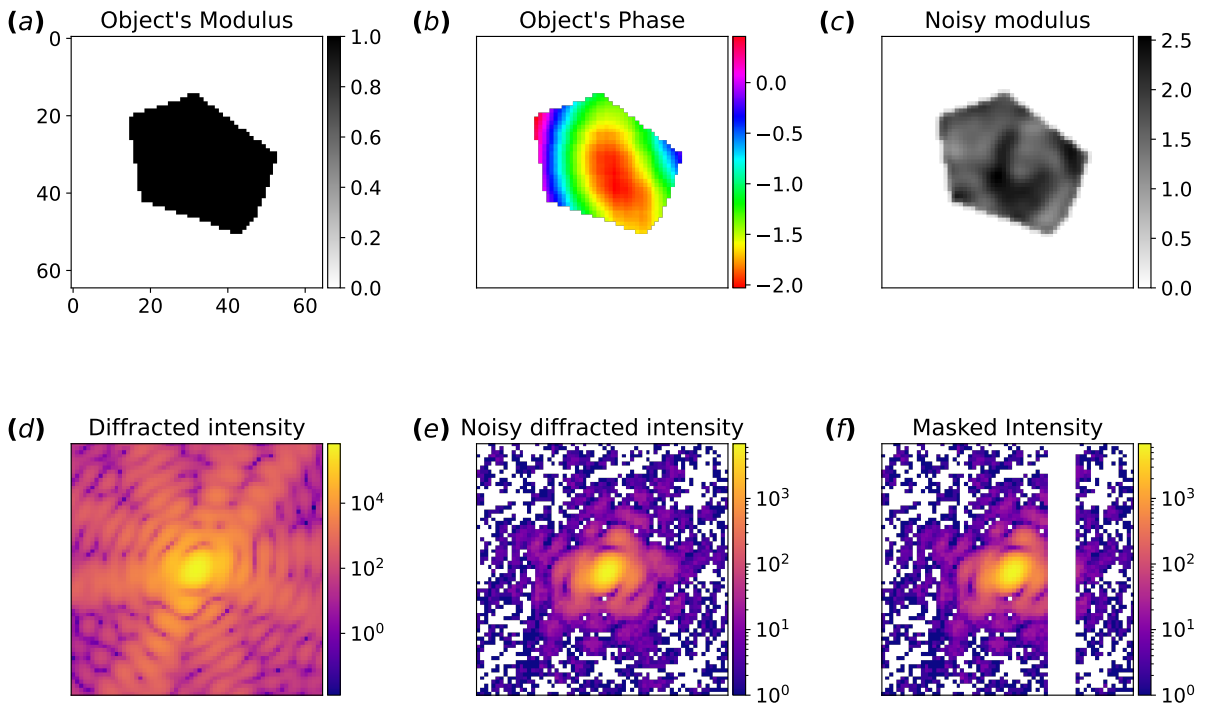


**Figure 4.3: Steps for the simulation of a single 2D diffraction pattern (a)** Simulated modulus of a 2D object with random shape and compact support. **(b)** Simulated object's phase **(c)** Object's modulus after smoothening the edges and adding random Gaussian noise. **(d)** Squared modulus of the Fourier Transform of the complex object (in log scale). The object is first padded with zeros to match the chosen oversampling ratio. **(e)** Poisson noise is added to the simulated diffracted intensity. **(f)** A 6 pixel-wide vertical gap is added to the diffracted intensity at a random position.

after the same training. Here in Fig. 4.4 we report the comparisons for the 9 pixel-wide gap on a test simulated diffraction pattern.

In the light of these results, we have decided to discard the MAE metric and adopt instead the sum of MSE and SSIM. At last, another term computing the MSE between the *gradients* of the ground truth and predicted intensity inside the gap region was added in the definitive loss function.

Once established what we considered the best loss function, we have explored different models. Following the work of Chavez *et al.* mentioned above ([16]), we considered a Mixed-Scale Dense Network (MSDnet). The advantage of this type of networks is the significant reduction of trainable parameters, and the use of *dilated* convolutions with respect to UNet ones. While the former property guarantees faster trainings and lower chances of overfitting, the latter enhances the capture of long-range correlations. Moreover, the image's spatial dimensions is kept constant throughout the whole network as no downsampling nor upsampling are operated. The MSDNet that we have used consists of sequential blocks in each of which the input is transformed by two different convolutional layers with growing dilation rates. Each output of the convolutional layers is concatenated to the input feature map and the result is passed to the following block. While the kernel size is kept constant to $3 \times 3 \times 3$ pixels the dilation rate increases linearly from 1 to 30. The last layer is a sigmoid function as well as for the UNet. The total number of trainable parameters is 320'000, significantly lower than the UNet.

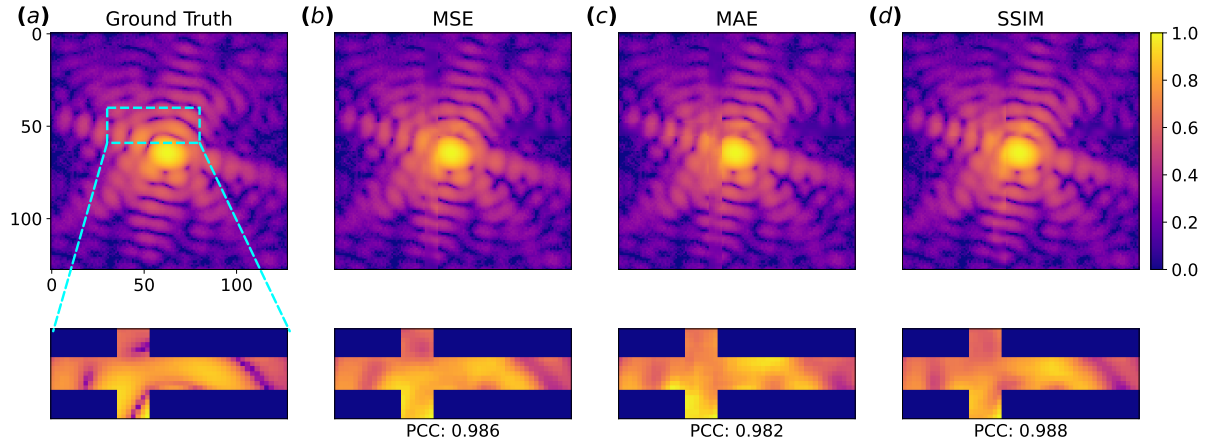


**Figure 4.4: Comparison of different losses** Results on a test simulated diffraction pattern for the inpainting of a 9 pixel-wide cross-shaped gap produced by the same UNet model trained for 10 epochs with different loss functions. **(a)** Shows the ground truth. **(b)** The prediction of the model trained with the MSE, **(c)** with the MAE, **(d)** with the SSIM. Corresponding accuracy scores calculated with the Pearson Correlation Coefficient (PCC) are shown as well. While MAE fails to recover the oscillations, SSIM yields better results.

In order to combine the hierarchical dimensionality reduction of the UNet with the fine-features capturing of the MSDNet we have implemented a modified UNet that adopts dilated convolutions inside the first two encoder blocks. In particular, they return the input tensor concatenated with the outputs of four dilated convolutional layers computed from the input. Dilation rates of (6,4,2,1) and (5,3,2,1) were chosen respectively. As the MaxPooling operation down-samples the feature maps into smaller sizes, we limited the dilated convolutions to the first two blocks. Moreover, we utilized them in the encoder layers only as they are mostly used for feature extraction [21].

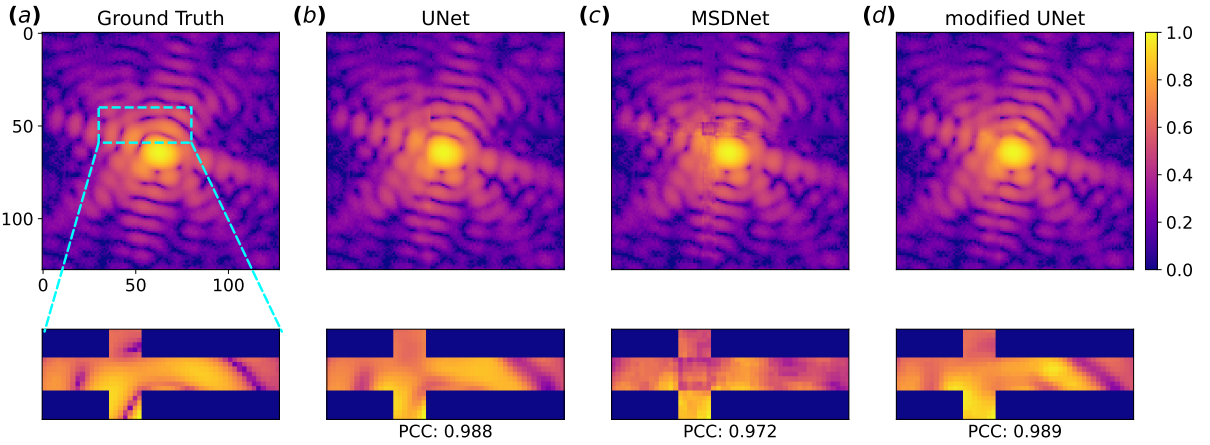


**Figure 4.5: Comparison of different models** Results on a test simulated diffraction pattern for the inpainting of a 9 pixel-wide cross-shaped gap using three different models trained with the same loss function. **(a)** Shows the ground truth. **(b)** The prediction of the UNet, **(c)** of the MSDNet, **(d)** of the modified UNet. Corresponding accuracy scores calculated with the Pearson Correlation Coefficient (PCC) are shown as well.

| | U-Net | MSD-Net | Unet_mod |
|---|---|---|---|
| block1 | `def encoder_block(x_input, num_filters, ker):`<br>`  s = Conv2D(num_filters, ker,' leaky_relu')(x_input)`<br>`  x = MaxPool2D(2)(s)`<br>`  return x, s` | `def MSD_block(x, in_channels, dilations,kernel_size=3):`<br>`  if isinstance(dilations, int):`<br>`  dilations = [(j % 10) + 1 for j in range( dilations)]`<br>`  out_channels = in_channels + len(dilations)`<br>`  for d in dilations:`<br>`  x1 = Conv2D(out_channels//2, kernel_size,1, dilation_rate=dilation, 'same', 'leaky_relu')( x)`<br>`  x = tf.concat([x1,x] ,axis = -1)`<br>`  return x, out_channels` | `def encoder_block_mod(x_input, ker , num_filters, rate):`<br>`  f = num_filters // 4`<br>`  s = tf.concat([x_input] + [ Conv2D(f, ker, dilation_rate=r, ' leaky_relu')(x_input) for r in rate], axis=-1)`<br>`  return MaxPool2D(2)(s), s` |
| block2 | `def decoder_block(x_input, num_filters, ker, skip_input = None):`<br>`  if skip_input is not None:`<br>`  x_input = Concatenate()([ x_input, skip_input])`<br>`  x = Conv2DTranspose(num_filters , ker, strides=2, ' leaky_relu')(x_input)`<br>`  return x` | | `def decoder_block(x_input, num_filters, ker, skip_input = None):`<br>`  if skip_input is not None:`<br>`  x_input = Concatenate()([ x_input, skip_input])`<br>`  x = Conv2DTranspose(num_filters , ker, strides=2, ' leaky_relu')(x_input)`<br>`  return x` |
| body | `x, s1 = encoder_block(inputs ,48,5)`<br>`x, s2 = encoder_block(x,96,3)`<br>`x, s3 = encoder_block(x,192,3)`<br>`x, s4 = encoder_block(x,384,3)`<br>`x = Conv2D(768,2,'leaky_relu')( x)`<br>`x = MaxPool2D(2)(x)`<br>`x = Conv2DTranspose(768,3,' leaky_relu')(x)`<br>`x = UpSampling2D(2)(x)`<br>`x = decoder_block(x,s4,384,3)`<br>`x = decoder_block(x,s3,192,3)`<br>`x = decoder_block(x,s2,96,3)`<br>`x = decoder_block(x,s1,48,4)`<br>`x = Conv2D(24,5,'leaky_relu')(x )`<br>`x = Conv2D(12,5,'leaky_relu')(x )`<br>`x = Conv2D(6,5,'leaky_relu')(x)`<br>`out = Conv2D(1,5,'sigmoid')(x)` | `x,out_ch = MSD_block(inputs ,1,[1,2])`<br>`x,out_ch = MSD_block(x,out_ch ,[3,4])`<br>`...`<br>`...`<br>`x,out_ch = MSD_block(x,out_ch ,[31,32])`<br>`out = Conv2D(1,3,'sigmoid')(x)` | `x, s1 = encoder_block_mod( inputs,3,48,[16,8,4,2])`<br>`x, s2 = encoder_block_mod(x,3, 96,[10,5,3,1])`<br>`x, s3 = encoder_block(x, 192, 3)`<br>`x, s4 = encoder_block(x, 384 ,3)`<br>`x, s5 = encoder_block(x, 768, 3)`<br>`x = Conv2D(1536,3,'leaky_relu') (x)`<br>`x = decoder_block(x,768,3)`<br>`x = decoder_block(x,384,3,s5)`<br>`x = decoder_block(x,192,3,s4)`<br>`x = decoder_block(x,96,3,s3)`<br>`x = decoder_block(x,48,4,s2)`<br>`x = Concatenate()([x, s1])`<br>`x = Conv2D(24,5,'leaky_relu')(x )`<br>`x = Conv2D(12,5,'leaky_relu')(x )`<br>`x = Conv2D(6,5,'leaky_relu')(x)`<br>`out = Conv2D(1,3,'sigmoid')(x` |
| parameters | 32'100'000 | 320'000 | 32'100'000 |

**Table 4.1:** Comparison of Unet, MSDNet, and Unet_mod components.

| | U-Net | MSD-Net | Unet_mod |
|---|---|---|---|
| **block1** | `def encoder_block(x_input, num_filters, ker):`<br>`  s = Conv2D(num_filters, ker,' leaky_relu')(x_input)`<br>`  x = MaxPool2D(2)(s)`<br>`  return x, s` | `def MSD_block(x, in_channels, dilations,kernel_size=3):`<br>`  if isinstance(dilations, int):`<br>`  dilations = [(j % 10) + 1 for j in range( dilations)]`<br>`  out_channels = in_channels + len(dilations)`<br>`  for d in dilations:`<br>`  x1 = Conv2D(out_channels//2, kernel_size,1, dilation_rate=dilation, 'same', 'leaky_relu')( x)`<br>`  x = tf.concat([x1,x] ,axis = -1)`<br>`  return x, out_channels` | `def encoder_block_mod(x_input, ker , num_filters, rate):`<br>`  f = num_filters // 4`<br>`  s = tf.concat([x_input] + [ Conv2D(f, ker, dilation_rate=r, ' leaky_relu')(x_input) for r in rate], axis=-1)`<br>`  return MaxPool2D(2)(s), s` |
| **block2** | `def decoder_block(x_input, num_filters, ker, skip_input = None):`<br>`  if skip_input is not None:`<br>`  x_input = Concatenate()([ x_input, skip_input])`<br>`  x = Conv2DTranspose(num_filters , ker, strides=2, ' leaky_relu')(x_input)`<br>`  return x` | | `def decoder_block(x_input, num_filters, ker, skip_input = None):`<br>`  if skip_input is not None:`<br>`  x_input = Concatenate()([ x_input, skip_input])`<br>`  x = Conv2DTranspose(num_filters , ker, strides=2, ' leaky_relu')(x_input)`<br>`  return x` |
| **body** | `x, s1 = encoder_block(inputs ,48,5)`<br>`x, s2 = encoder_block(x,96,3)`<br>`x, s3 = encoder_block(x,192,3)`<br>`x, s4 = encoder_block(x,384,3)`<br>`x = Conv2D(768,2,'leaky_relu')( x)`<br>`x = MaxPool2D(2)(x)`<br>`x = Conv2DTranspose(768,3,' leaky_relu')(x)`<br>`x = UpSampling2D(2)(x)`<br>`x = decoder_block(x,s4,384,3)`<br>`x = decoder_block(x,s3,192,3)`<br>`x = decoder_block(x,s2,96,3)`<br>`x = decoder_block(x,s1,48,4)`<br>`x = Conv2D(24,5,'leaky_relu')(x )`<br>`x = Conv2D(12,5,'leaky_relu')(x )`<br>`x = Conv2D(6,5,'leaky_relu')(x)`<br>`out = Conv2D(1,5,'sigmoid')(x)` | `x,out_ch = MSD_block(inputs ,1,[1,2])`<br>`x,out_ch = MSD_block(x,out_ch ,[3,4])`<br>`...`<br>`...`<br>`x,out_ch = MSD_block(x,out_ch ,[31,32])`<br>`out = Conv2D(1,3,'sigmoid')(x)` | `x, s1 = encoder_block_mod( inputs,3,48,[16,8,4,2])`<br>`x, s2 = encoder_block_mod(x,3, 96,[10,5,3,1])`<br>`x, s3 = encoder_block(x, 192, 3)`<br>`x, s4 = encoder_block(x, 384 ,3)`<br>`x, s5 = encoder_block(x, 768, 3)`<br>`x = Conv2D(1536,3,'leaky_relu') (x)`<br>`x = decoder_block(x,768,3)`<br>`x = decoder_block(x,384,3,s5)`<br>`x = decoder_block(x,192,3,s4)`<br>`x = decoder_block(x,96,3,s3)`<br>`x = decoder_block(x,48,4,s2)`<br>`x = Concatenate()([x, s1])`<br>`x = Conv2D(24,5,'leaky_relu')(x )`<br>`x = Conv2D(12,5,'leaky_relu')(x )`<br>`x = Conv2D(6,5,'leaky_relu')(x)`<br>`out = Conv2D(1,3,'sigmoid')(x)` |
| **parameters** | 32'100'000 | 320'000 | 32'100'000 |

**Table 4.2:** Comparison of Unet, MSDNet, and Unet_mod components.

# Chapter 5

# Deep Learning for Phase Retrieval

We enter now the core topic of the thesis. Most of the efforts during this PhD have been dedicated to the Phase Problem.

## 5.1 State of the art

## 5.2 Highly strained crystals

## 5.3 Reciprocal space phasing

## 5.4 Phase symmetries breaking

## 5.5 Model design

## 5.6 Results on 2D case

## 5.7 Results on 3D case

## 5.8 Refinement with iterative algorithms

## 5.9 Experimental results

# Chapter 6

# Conclusions

# Appendix A

# Additional Data and Methods

Appendix B

# Appendix

# Bibliography


1. Ponchut, C. *et al. MAXIPIX, a fast readout photon-counting X-ray area detector for synchrotron applications* in *Journal of Instrumentation* **6.** Issue: 1 ISSN: 17480221 (Jan. 2011).
2. Johnson, I. *et al.* Eiger: a single-photon counting x-ray detector. *Journal of Instrumentation* **9,** C05032. https://dx.doi.org/10.1088/1748-0221/9/05/C05032 (May 2014).
3. Carnis, J. *et al.* Towards a quantitative determination of strain in Bragg Coherent X-ray Diffraction Imaging: artefacts and sign convention in reconstructions. *Scientific Reports* **9.** Publisher: Nature Research. ISSN: 20452322 (Dec. 1, 2019).
4. Elharrouss, O., Almaadeed, N., Al-Maadeed, S. & Akbari, Y. Image Inpainting: A Review. *Neural Processing Letters* **51,** 2007–2028. ISSN: 1573-773X. http://dx.doi.org/10.1007/s11063-019-10163-0 (Dec. 2019).
5. Jam, J. *et al.* A comprehensive review of past and present image inpainting methods. *Computer Vision and Image Understanding* **203,** 103147. ISSN: 1077-3142. https://www.sciencedirect.com/science/article/pii/S1077314220301661 (2021).
6. Efros, A. & Leung, T. *Texture synthesis by non-parametric sampling* in *Proceedings of the Seventh IEEE International Conference on Computer Vision* **2** (1999), 1033–1038 vol.2.
7. Bertalmio, M., Bertozzi, A. & Sapiro, G. *Navier-stokes, fluid dynamics, and image and video inpainting* in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001* **1** (2001), I–I.
8. Mairal, J., Elad, M. & Sapiro, G. Sparse Representation for Color Image Restoration. *IEEE Transactions on Image Processing* **17,** 53–69 (2008).
9. Allene, C. & Paragios, N. *Image Renaissance Using Discrete Optimization* in *18th International Conference on Pattern Recognition (ICPR'06)* **3** (2006), 631–634.
10. Goodfellow, I. J. *et al. Generative Adversarial Networks* 2014. arXiv: 1406.2661 [stat.ML]. https://arxiv.org/abs/1406.2661.
11. Jiang, Y., Xu, J., Yang, B., Xu, J. & Zhu, J. Image Inpainting Based on Generative Adversarial Networks. *IEEE Access* **8,** 22884–22892 (2020).
12. Xu, Z. *et al.* A Review of Image Inpainting Methods Based on Deep Learning. *Applied Sciences* **13.** ISSN: 2076-3417. https://www.mdpi.com/2076-3417/13/20/11189 (2023).
13. Li, C.-T. *10 Papers You Must Read for Deep Image Inpainting* Accessed: 2025-03-03. 2020. https://towardsdatascience.com/10-papers-you-must-read-for-deep-image-inpainting-2e41c589ced0/.
14. Sogancioglu, E., Hu, S., Belli, D. & van Ginneken, B. *Chest X-ray Inpainting with Deep Generative Models* 2018. arXiv: 1809.01471 [cs.GR]. https://arxiv.org/abs/1809.01471.
15. MA, B. *et al.* Deep learning-based automatic inpainting for material microscopic images. *Journal of Microscopy* **281,** 177–189. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/jmi.12960. https://onlinelibrary.wiley.com/doi/abs/10.1111/jmi.12960 (2021).

16. Chavez, T., Roberts, E. J., Zwart, P. H. & Hexemer, A. A comparison of deep-learning-based inpainting techniques for experimental X-ray scattering. *Journal of Applied Crystallography* **55.** Publisher: International Union of Crystallography (IUCr), 1277–1288. ISSN: 1600-5767. https://scripts.iucr.org/cgi-bin/paper?S1600576722007105 (Oct. 1, 2022).

17. Bellisario, A., Maia, F. R. & Ekeberg, T. Noise reduction and mask removal neural network for X-ray single-particle imaging. *Journal of Applied Crystallography* **55.** Publisher: International Union of Crystallography, 122–132. ISSN: 16005767 (Feb. 1, 2022).

18. Lecun, Y., Bottou, L., Orr, G. & Müller, K.-R. Efficient BackProp (Aug. 2000).

19. Wang, Z., Bovik, A., Sheikh, H. & Simoncelli, E. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* **13,** 600–612 (2004).

20. Chen, L.-C., Papandreou, G., Schroff, F. & Adam, H. Rethinking Atrous Convolution for Semantic Image Segmentation. arXiv: 1706.05587. http://arxiv.org/abs/1706.05587 (June 17, 2017).

21. Yu, F. & Koltun, V. *Multi-Scale Context Aggregation by Dilated Convolutions* 2016. arXiv: 1511.07122 [cs.CV]. https://arxiv.org/abs/1511.07122.

# Annexes

Appendix A

# Appendix