Tomography of Materials and Structures 4 (2024) 100019

Contents lists available at ScienceDirect

Tomography of Materials and Structures

journal homepage: www.journals.elsevier.com/tomography-of-materials-and-structures

Conditional generative adversarial networks for stripe artefact removal in high-resolution X-ray tomography

Daniil Kazantsev [a,*], Lucas Beveridge [a,b], Vigneshwar Shanmugasundar [a], Oxana Magdysyuk [c]

[a] Diamond Light Source Ltd. Diamond House, Harwell Science and Innovation Campus, Fermi Ave, Didcot OX11 0DE, UK
[b] The University of Birmingham, Edgbaston, Birmingham B15 2TT, UK
[c] University of St. Andrews, KY16 9ST, UK

ARTICLE INFO

Keywords:
Image reconstruction
Ring removal
Sinogram inpainting
Data extrapolation
Deep learning
GAN
Material science
Synchrotron
X-ray tomography

ABSTRACT

Tomographic imaging supports a great number of medical and material science applications. The collected projection data usually has different types of imaging artefacts and noise. Various image pre-processing and reconstruction methods are used to obtain volumetric datasets of high quality for further analysis. In order to minimise reconstruction artefacts, one can apply either filtering and/or data completion/inpainting techniques which can recover the data. Deep learning (DL) methods to remove artefacts and noise have been successfully applied in the past. In this paper, we present a novel approach based on conditional generative adversarial networks (cGANs) to remove stripe artefacts. The novelty of the presented technique is in how the training data for DL is extracted from the same tomographic dataset that needs recovery. We also provide new deterministic stripe detection and inpainting algorithms to support the development. The presented methods are compared with other stripe removal algorithms and applied to 3D and 4D high-resolution X-ray data collected at Diamond Light Source synchrotron, UK. The proposed DL method delivers reconstructed images with minimised ring artefacts while being a parameter-free approach. A similar DL strategy can also be applied to remove other types of artefacts in images.

1. Introduction

X-ray computed tomography (XCT) is a versatile non-invasive imaging technique which supports many medical and material science applications [1,2]. The main physical principle of XCT is to collect 2-dimensional (2D) radiographic projections by rotating the gantry of a scanner while keeping the scanned object fixed. Alternatively, one can rotate the object itself while keeping the detectors fixed, which is a set-up for synchrotron X-ray imaging.

After a series of image radiographs/projections are collected, the three dimensional (3D) inner structure of an object can be resolved by using a reconstruction algorithm. Commonly used direct reconstruction methods include the Filtered Back Projection (FBP) algorithm for parallel and fan-beam geometries, and the Feldkamp-Davis-Kress algorithm for cone-beam geometry [2].

Unfortunately, direct reconstruction techniques are not flexible enough to compensate for the variety of imaging artefacts and noise, which are a part of the collected data. This frequently results in various distortions in the reconstructed images due to unrealistic assumptions of the inversion model used for reconstruction [3].

There are two main strategies to deal with the problem of tomographic data being inaccurate. The first approach is the pre-processing of the projection data before applying the direct reconstruction. The second is to use a more realistic mathematical model as a part of an iterative reconstruction method. Both approaches have their own advantages and disadvantages as it is shown in Table 1.

Although iterative methods usually produce better quality reconstruction results as they rely on more rigorous mathematical models, they can be impractical for big data applications.

In this paper, we focus on a series of data pre-processing strategies to eliminate certain artefacts that are routinely present in synchrotron X-ray data [4]. Stripe artefacts in projection data (in sinogram space) could be a result of different physical phenomena that affect the acquisition hardware. The list of possible factors is extensive, but the major contributor is usually scintillator defects [5]. Stripe artefacts contribute to the inconsistency of the collected data, and direct reconstruction will result in ring artefacts of different prominence in the reconstructed images. Iterative methods can minimise the ring artefacts by

* Corresponding author.
E-mail address: daniil.kazantsev@diamond.ac.uk (D. Kazantsev).

https://doi.org/10.1016/j.tmater.2023.100019
Received 1 September 2023; Received in revised form 9 November 2023; Accepted 16 November 2023
Available online 24 November 2023

Table 1
Cons and pros of the data pre-processing and direct reconstruction compared to iterative reconstruction.

	Pre-processing/Direct recon.	Iterative reconstruction
Pros:	Computationally efficient Practical for big data More controllable Less parameters involved	Mathematically accurate A higher quality solution Versatile
Cons:	Heuristic Raw data modification	Computationally challenging More parameters involved Fine tuning is problematic

incorporating a more complex model into the objective, but this usually results in a more complicated algorithm which is difficult to optimise and tune [6–8].

Recently, generative adversarial networks (GANs) have been successfully used for the more general problem of sinogram inpainting/data completion in tomography [9–12,13]. In our study, we mainly focus on a type of stripe artefact ('unresponsive' or 'dead' [5]) that suppresses all information about the underlying data, therefore the data needs to be recovered by data completion algorithms. Here we use a DL method based on a conditional generative adversarial network (cGAN) [14] to perform inpainting of the missing data in the stripe regions only. The conditioning of GANs is performed by knowing the exact location of an artefact in the image. This is implemented by using a novel stripe detection algorithm which is also presented.

Since cGANs are a supervised DL technique, they require labelled training data. Having representative and generic training data can be a major hurdle in applying a DL method effectively. Here, we present a novel approach for extracting training data from the acquired raw data. This ensures a close proximity of the feature space of the trained model to the test data. To avoid over-fitting, we dilute the training data with data from different samples and demonstrate that the model can generalise well when applied to unseen data.

We compare the presented DL technique and another novel inpainting method with two popular stripe removal filters. Notably, the majority of filters act on the whole image globally to remove an artefact, while we target only local areas where artefacts may be present. This minimises raw data changes which can lead to various distortions in the reconstructed images.

The methods were applied to 3D and 4D (dynamic) real data collected at Diamond Light Source.

2. Methods

In this section, we present novel deterministic algorithms for stripe detection and data inpainting, as well as the neural network methodology to perform data completion for missing data in stripes.

2.1. Stripes detection approach

In order to acquire the training dataset for the neural network, we need to establish where stripes are located in the data. This is needed to: a) identify stripe-free regions in the raw data; b) use stripe-free regions to simulate synthetic stripes and obtain targets and inputs for the training dataset; c) remove true stripes which were detected in the data.

There are stripe-removal algorithms that use detection before applying a filter to minimise the artefacts [5,15]. Frequently, the detection problem is not decoupled from the filtering process and therefore it is hard to evaluate how well the detection algorithm performed. It is also critical to reduce the level of false alarms and therefore modifications of the raw data in areas where no artefacts are present. The detection methods which rely on averaging intensity along the angular axis in sinograms and detecting peaks in 1D signals [5] are too sensitive to various stripe-like features and frequently cannot differentiate between partial and full stripes.

In order to establish a more robust and controlled stripe detection process, we have developed a novel stripe-detection algorithm. It is a non-local 3D method which helps to reduce the level of false alarms during the stripe detection process. The method can differentiate between different types of stripes as well as features that belong to data and should be excluded. It is a versatile and parametrised algorithm which gives control over the sensitivity of a detection process to a user.

The proposed method consists of two main parts: .

1. Calculation of weights in the gradient space of the sinogram.
2. Weights thresholding subject to morphological constraints to establish a binary mask.

In step 1, we calculate an image gradient using forward differences in x-direction as $\nabla_x g(\theta, x, y) = g(\theta, x+h, y) - g(\theta, x, y)$ of normalised 3D projection data $g(\theta, x, y)$. Here (x, y) coordinates are the horizontal and vertical coordinates of the 2D detector plane, respectively. In sinogram space, chosen x-direction aligns with the horizontal detector axis and orthogonal to the angular θ-axis.

For every element of the gradient $\nabla_x g(\mathbf{x})$, $\mathbf{x} = (\theta, x, y)$, we apply a specially designed 3D stencil centred around voxel $\mathbf{x}$, depicted in Fig. 1. We define a mean $\mu(\nabla_x g)_{\Omega_{A,B,C}}$ for A, B and C stencil regions respectively and calculate the following weights:

$$\omega_1(\mathbf{x}) = \begin{cases} \dfrac{|\mu(\nabla_x g)_{\Omega_A}|}{|\mu(\nabla_x g)_{\Omega_B}|}, & \text{if } |\mu(\nabla_x g)_{\Omega_A}| \le |\mu(\nabla_x g)_{\Omega_B}| \\ \dfrac{|\mu(\nabla_x g)_{\Omega_B}|}{|\mu(\nabla_x g)_{\Omega_A}|}, & \text{otherwise.} \end{cases} \tag{1}$$

$$\omega_2(\mathbf{x}) = \begin{cases} \dfrac{|\mu(\nabla_x g)_{\Omega_A}|}{|\mu(\nabla_x g)_{\Omega_C}|}, & \text{if } |\mu(\nabla_x g)_{\Omega_A}| \le |\mu(\nabla_x g)_{\Omega_C}| \\ \dfrac{|\mu(\nabla_x g)_{\Omega_C}|}{|\mu(\nabla_x g)_{\Omega_A}|}, & \text{otherwise.} \end{cases} \tag{2}$$

and the final weight is calculated as $\omega(\mathbf{x}) = \min(\omega_1(\mathbf{x}), \omega_2(\mathbf{x}))$, and the obtained weights are in the range $\boldsymbol{\omega} \in (0, 1]$, where smaller weights represent more prominent stripes in the data.

In order to minimise the amount of false alarms detected, we further apply a 1D median filter to ω in θ-direction, with a kernel size that depends on the length of stripes we expect to have in the data.

The general idea for this method is to exploit the fact that the gradient jump in x-direction should be significantly larger than in

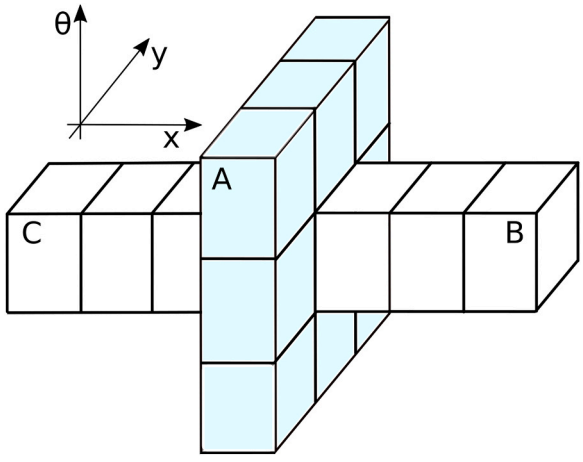

Fig. 1. 3D detection stencil which consists of a 2D 3 × 3 kernel A, which is parallel to θ-direction and two 1D 1 × 3 kernels B and C, which are parallel to detector's x-direction. The stencil is centred around a voxel $\mathbf{x}$ at the centre of the stencil A (hidden).

θ-direction when the stripe is present. The non-local nature of the presented stencil (see Fig. 1) ensures a robustness to noise for more accurate edge detection. The result of applying this method can be seen in Fig. 2 (middle). See that the detected stripes have weights around 0.5 while the rest of the data is more than 0.6.

In the step 2 of the detection method, in order to obtain a binary mask $\mathscr{M}(\mathbf{x})$, we process the estimated weights $\omega(\mathbf{x})$ by applying a thresholding process with additional morphological constraints. In Fig. 2 (middle) one can see that there are vertical features that belong to stripes but also to sample data (e.g. two features at the bottom right section of the image). The aim is to build a mask that would only have stripe artefacts and useful data features are excluded, meaning that we cannot accept any prominent vertical boundary to be a stripe.

The method in step 2 checks the consistency of weights $\omega(\mathbf{x})$ in three dimensions in order to generate a mask $\mathscr{M}(\mathbf{x})$. We parametrise stripes by having geometrical constraints for length, width and depth (L, W, D). We assume that stripes do not usually extend deep (D constraint) compared to sample features and also can be reasonably lengthy (L constraint). The stripes are also usually not very thick (W constraint).

First we test every weight element $\omega(\mathbf{x})$ that it is bellow a given threshold T. Then we test if the geometrical constraints above hold for the weight candidate and reject it if they are not met (3). The L, W, D parameters can be roughly estimated from the dimensions of the input data and the type of stripes present. Once they are established, they usually remain fixed for other datasets with the same dimensions and similar stripe features. For our experiments with data dimensions $1801 \times 2160 \times 2560$ voxels, we fixed them as $L = 600$; $W = 22$; $D = 33$ in pixel units. While accepting insignificant deviations from the L, W, D values (sensitivity), we reject features that are shorter than L, thicker than W and deeper than D based on the obtained weights $\omega(\mathbf{x})$ (3).

$$\mathscr{M}(\mathbf{x}) = \begin{cases} 1, & \text{if } \omega(\mathbf{x}) \leq T \text{ s.t. geometrical constraints : } L, W, D \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Here T is a user defined threshold which controls the sensitivity of the detection. Usually the values between 0.5 and 0.7 provide good results; we used $T = 0.63$ in all our experiments.

Fig. 2 (right) shows the resulting binary mask where one full and one partial stripe artefacts detected. Notably all other features, even those with the weights bellow the threshold T, have been excluded as they did not satisfy the imposed geometrical constraints.

We tested this method extensively on different samples when generating training data for the DL inpainting approach (see Sec. 3.2). The method is proven to be reliable to detect different types of stripes in the data with a low level of false alarms.

The presented method for stripes detection and mask generation is implemented in C language and integrated into open source TomoPy package with Python interface [16].

2.2. Data inpainting method

Many inpainting techniques are based on various diffusion models, i. e. a subject to a solution of a partial differential equation (PDE) [17,18]. The PDE-based methods, however, frequently blur the inpainted region and also require implementing non-trivial iterative numerical schemes to ensure convergence stability. The expensive iterative PDE evolution can be also time-consuming and impractical for large data sizes in material science. Some inpainting methods try to predict the direction of smoothing [19,20], but for complex samples this is rarely possible.

The proposed inpainting method was designed with several targets in mind: effectiveness, stability, minimum of free parameters and computational feasibility for big data. It is a morphological approach, similar to methods in [21,22].

Let Ω be the region to be inpainted in the sinogram $g(\theta, x, y)$ and $\partial\Omega$ defines a one-voxel thick boundary between the missing data and the available data (see Fig. 3). For stripe artefacts, the $\partial\Omega$ boundaries are located on the opposite sides of the inpainted region Ω. In iterations, the boundary $\partial\Omega$ gets updated with new values and the front propagates until it merges with another boundary.

We define a 3D non-local searching window $N(\boldsymbol{x})$ for each boundary voxel such that $\boldsymbol{x} \in \partial\Omega$ as $(2\mathscr{N}_s(\boldsymbol{x}) + 1)^3$, where $\mathscr{N}_s$ is the half-width of the window. The inpainting algorithm propagates a random element $N(\boldsymbol{x}') \notin \Omega$ into the point $\boldsymbol{x} \in \partial\Omega$. It is also possible to propagate mean or median of values in the neighbourhood $N(\boldsymbol{x})$ (implemented in the

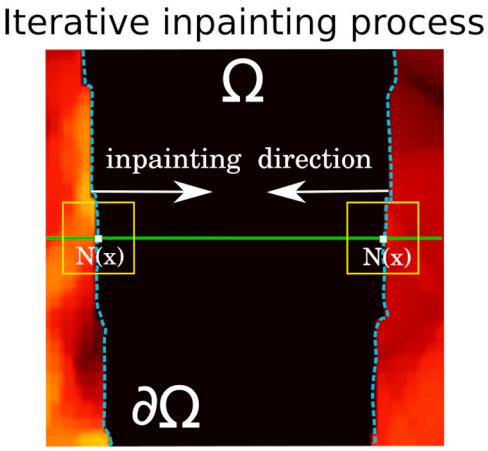

Fig. 3. The image shows the process of inpainting the missing data in the Ω stripe region. The boundaries $\partial\Omega$ are located on the opposite sides of Ω and they contract in iterations while getting inpainted. Non-local window $N(\boldsymbol{x}) \in \partial\Omega$ defines the area where a random element $N(\boldsymbol{x}') \notin \Omega$ is selected for inpainting.

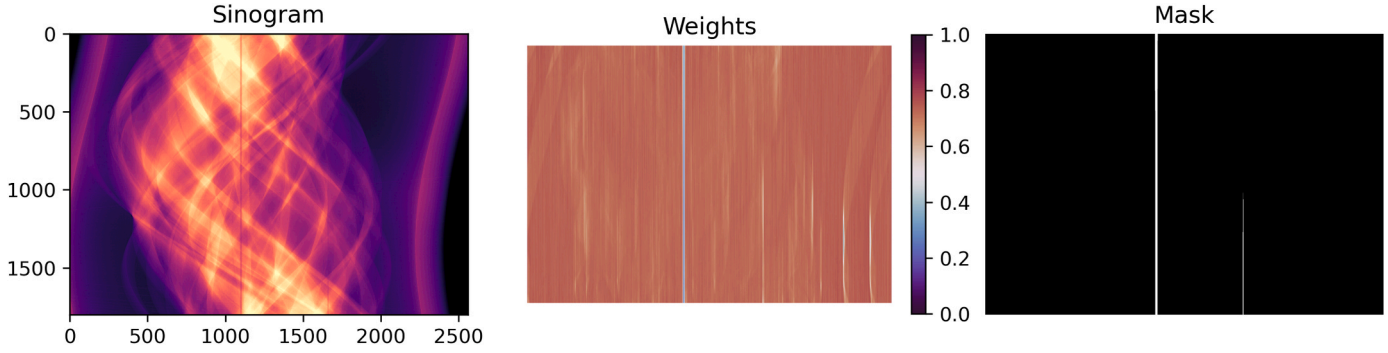

Fig. 2. The results of applying stripe detection algorithm to sinogram data. Left image shows a sinogram with two stripes present, one full (around 1100 on the horizontal x-axis) and one partial and weaker (around 1650); Middle image shows the result of applying step 1 to calculate weights; Right image demonstrates the binary mask obtained from the weights. Note that this detection method was able to detect artefacts of different length and intensity and reject false alarms (features that belong to the sample) at the same time.

method), however, for this study we used only the random assignment method.

The proposed approach has only two free parameters $\mathscr{N}_s$ - the half-size of the searching window and the number of iterations to raster through the already inpainted area (relates to the smoothing effect). For our experiments we used $\mathscr{N}_s = 5$ and the number of iterations equals to 5. The window $\mathscr{N}_s$ should be relatively small to avoid $N(\boldsymbol{x})$ containing neighbours that are too distant (see Sec. 5).

To minimise the presence of outliers that can be generated by random selection, we apply a Gaussian filter after every iteration of the algorithm.

Overall the proposed technique is simple and fast, yet generally effective, as we show in Sec. 4. The main idea of the method is to sample values from the neighbourhood of usable data. This method reduces the amount of blurring and regions overspilling compared to inpainting methods based on PDEs (e.g. linear or non-linear diffusion) or averaging (see Fig. 4).

In Fig. 4, the morphological inpainting based on the mean value results in a blurred area with a significant level of intensity overspilling (see the diagonal edge area where brighter intensities merged into darker ones). The median-based inpainting reduces the overspilling, but introduces the vertical artefact where contracting boundaries meet during inpainting process. This can be minimised by using larger kernels $\mathscr{N}_s \geq 7$, but the computational speed will decrease substantially. The proposed method based on the random value shows more naturally looking inpainted region with less overspill. The inpainting with cGAN shows an accurate recovery with the direction of features preserved and without intensity spill. It is also the most natural looking inpainted area without smoothing.

2.3. Conditional generative adversarial networks

2.3.1. Objective function

GANs were first proposed in [23] as a minimax game consisting of two players; a Generator G and a Discriminator D. The goal of the Discriminator is to classify images as either real or generated. The goal of the Generator is to fool the Discrminator into classifying its generated data as real. Formally, this can be written as:

$$\min_G \max_D \mathscr{J}_{GAN}(G,D) = \mathbb{E}_y[\log D(y)] + \mathbb{E}_z[\log(1 - D(G(z)))] \tag{4}$$

where z is a vector of random noise and y is a sample from the real data distribution.

Conditional GANs [24] are a variant of GANs that involve conditioning the Generator and Discriminator on some extra information x:

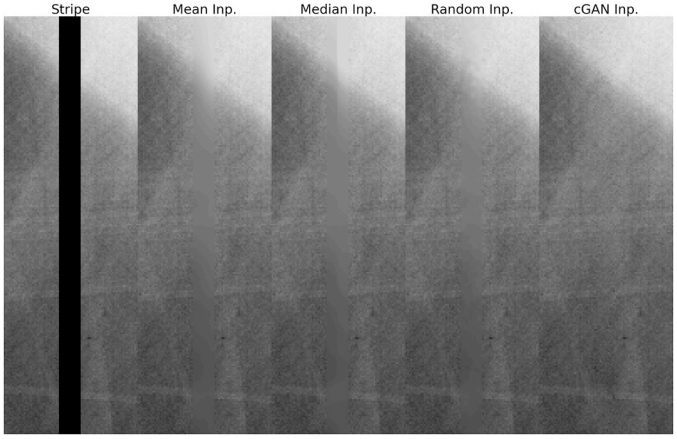

Fig. 4. Applying different types of morphological inpainting and the proposed cGAN inpainting to a missing data region in the sinogram. Notice the natural looking inpainted region while using cGAN compared to other methods.

$$\min_G \max_D \mathscr{J}_{GAN}(G,D) = \mathbb{E}_{x,y}[\log D(x,y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x,z)))] \tag{5}$$

Following the approach of [14], we do not provide G with the random noise vector z, instead providing G solely with the conditional information x. This is because for the problem of ring artefact removal, a machine learning method should be as deterministic as possible. Any randomness in the Generator's input could lead to randomness in the inpainted stripes, which in turn could cause artefacts in the reconstructions. Removing z decreases the stochasticity of the network. So we re-write Eq. (5) as:

$$\min_G \max_D \mathscr{J}_{GAN}(G,D) = \mathbb{E}_{x,y}[\log D(x,y)] + \mathbb{E}_{x}[\log(1 - D(x, G(x)))] \tag{6}$$

In our case, x is a sinogram with stripe artefacts, and y is the same sinogram but without stripes (a target). In theory, given x, G will learn to generate sinograms that look as close to y as possible. Practically, however, this is not always true. GANs are notoriously difficult to train, and often lead to problems such as mode collapse or early convergence [25], affecting the quality of generated images.

Following the approach of [14], we include an L^1 loss term in the objective of G to stabilise training and to encourage generated images to look similar to real images in an absolute sense:

$$\mathscr{J}_{L^1}(G) = \mathbb{E}_{x,y}[\| y - G(x) \|_1] \tag{7}$$

The objective of the GAN now becomes:

$$\min_G \max_D \mathscr{J}_{GAN}(G,D) + \lambda_{L^1} \min_G \mathscr{J}_{L^1}(G) \tag{8}$$

where λ_{L^1} is a weighting hyperparameter.

Preliminary testing showed that networks struggled to generate sinogram images with the resolution required for a high quality reconstruction. To avoid this problem, a binary mask $\mathscr{M}$ (3) is used to indicate the locations of stripe artefacts, where $1 =$ *stripe* and $0 =$ *no stripe*.

Rather than conditioning the GAN on the sinogram with stripes x, we condition it on the mask $\mathscr{M}_x$. Additionally, as the non-stripe areas of the sinogram have no artefacts and do not need changing, we add a final post-processing step to G so that only stripe areas of the input are changed. The final objective with these changes is as follows:

$$\min_{\widehat{G}} \max_D \mathscr{J}_{GAN}(\widehat{G},D) + \lambda_{L^1} \min_{\widehat{G}} \mathscr{J}_{L^1}(\widehat{G}) \tag{9}$$

where

$$\widehat{G}(x, \mathscr{M}_x) = (1 - \mathscr{M}_x) \odot x + \mathscr{M}_x \odot G((1 - \mathscr{M}_x) \odot x), \tag{10}$$

and $\odot$ is the component-wise multiplication.

The process of conditioning generator G on a binary mask $\mathscr{M}_x$ is shown in the Fig. 5 and the training process in Fig. 6.

We also re-formulate the L^1 loss term so that it is only calculated on stripe regions of the generated image. As shown in Fig. 5 and formulated in Eq. (10), non-stripe regions in the generated image get replaced by the corresponding non-stripe regions in the input image. As the input is identical to the target apart from the stripes, the absolute error on non-stripe regions will be 0. This will skew the L^1 loss when averaged across every pixel, and so the Generator will suffer from vanishing gradients. Therefore, in the backward pass, G's weights should only be updated with respect to their contribution to stripe regions.

The mask-conditioned L^1 loss can be expressed as:

$$\mathscr{J}_{L^1}(G, \mathscr{M}_x) = \mathbb{E}_{x,y}[\| (\mathscr{M}_x \odot y) - (\mathscr{M}_x \odot G(x)) \|_1] \tag{11}$$

2.3.2. Network architecture

We apply our models to patches of sinograms of size 1801×256, rather than whole sinograms. This splitting ensures that the angular dimension where the stripe lies is full and not cropped. Notably, this is different to the PatchGAN approach of [14], where they use the

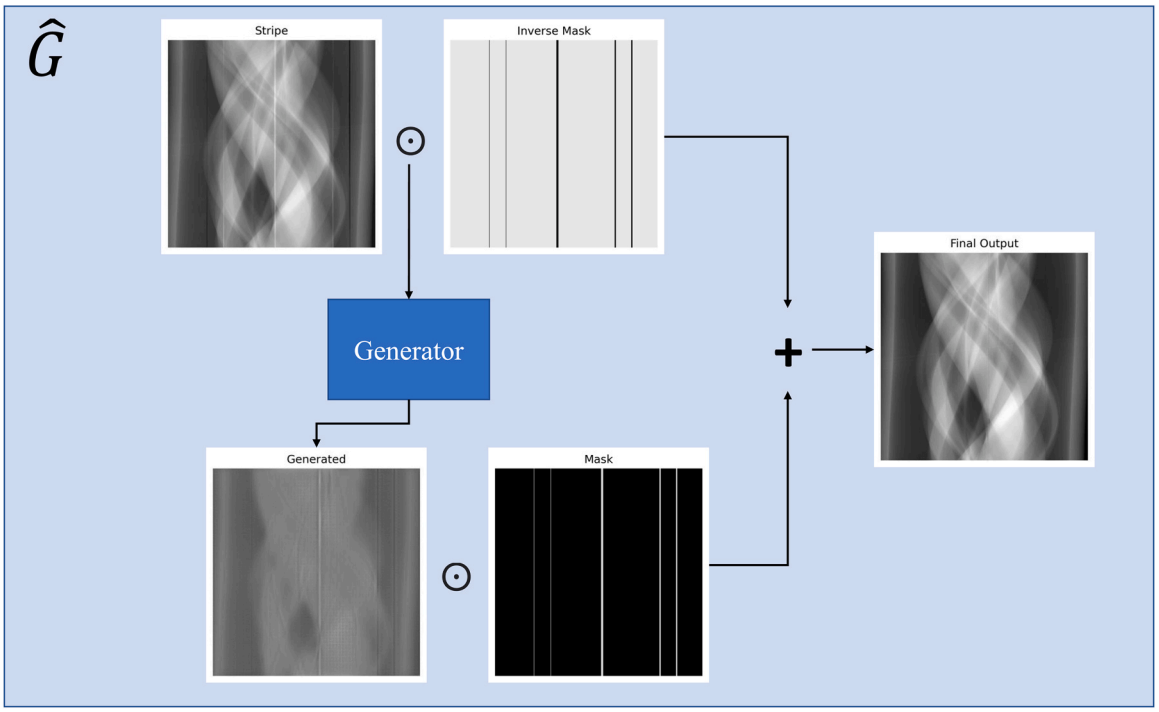

Fig. 5. Diagram showing how the Generator is conditioned on a binary mask ($\widehat{G}$ as described in Eq. (10)). Here *Stripe* is a sinogram with stripes, *Mask* is a binary mask indicating the locations of those stripes, and *Inverse Mask* is the negation of *Mask*. The symbol $\odot$ represents element-wise multiplication, and $+$ represents element-wise addition.

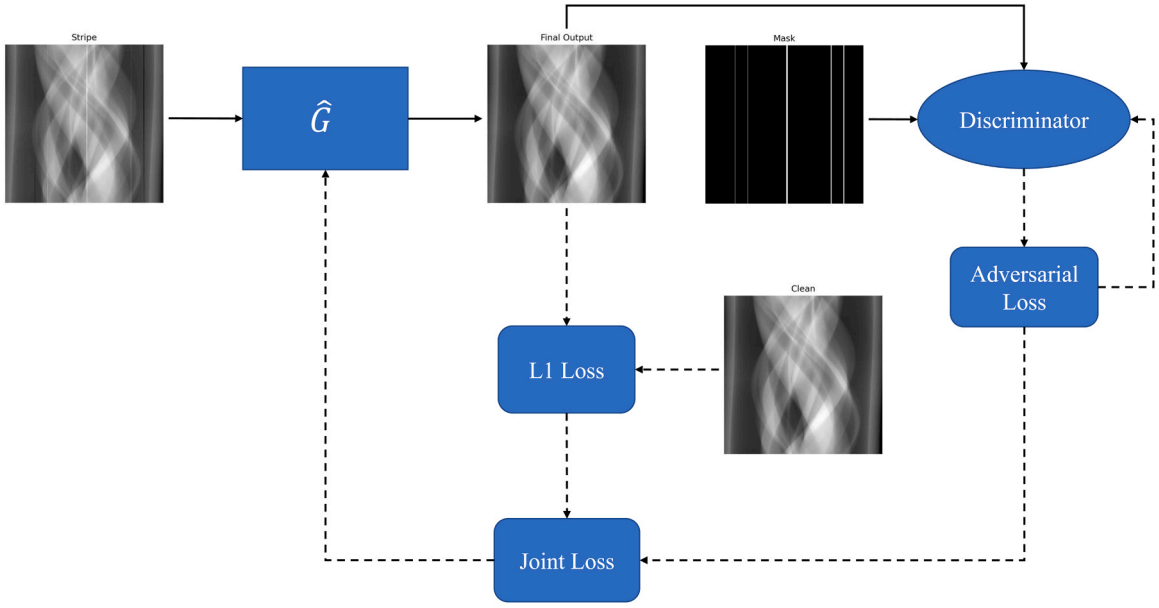

Fig. 6. The training procedure for objective function optimisation. Here *Clean* is the same sinogram as *Stripe* without stripes. *Joint Loss* represents the weighted summation of adversarial and L^1 loss in Eq. (9). Solid lines indicate the forward pass, dotted lines indicate the backward pass.

receptive field of the discriminator to virtually split an input image into patches. We split every image in the dataset into patches as a pre-processing step, and design our network architecture around this.

We used a U-Net generator [26] with eight encoding layers and eight decoding layers (see the top image in Fig. 7). Each encoding layer consists of a 2D convolution with kernel size 4, stride 2 and padding 1, followed by batch normalisation with $\epsilon = 0.001$, and finally a leaky ReLU with slope 0.2. The first and final decoding layers don't include batch normalisation, and the final layer has a normal (non-leaky) ReLU activation. The first encoding layer has 64 convolutional filters, which doubles every layer down to layer 4, where it remains at 512 until the final encoding layer. Each decoding layer consists of a 2D transposed convolution with kernel size 4, stride 2 and padding 1, followed by batch normalisation with $\epsilon = 0.001$, and finally a ReLU activation. The final decoding layer doesn't include batch normalisation, and uses a tanh activation. Additionally, the first three decoding layers include dropout with $p = 0.5$. Skip connections concatenate the output of an encoding layer with a decoding layer, which is then input to the next decoding

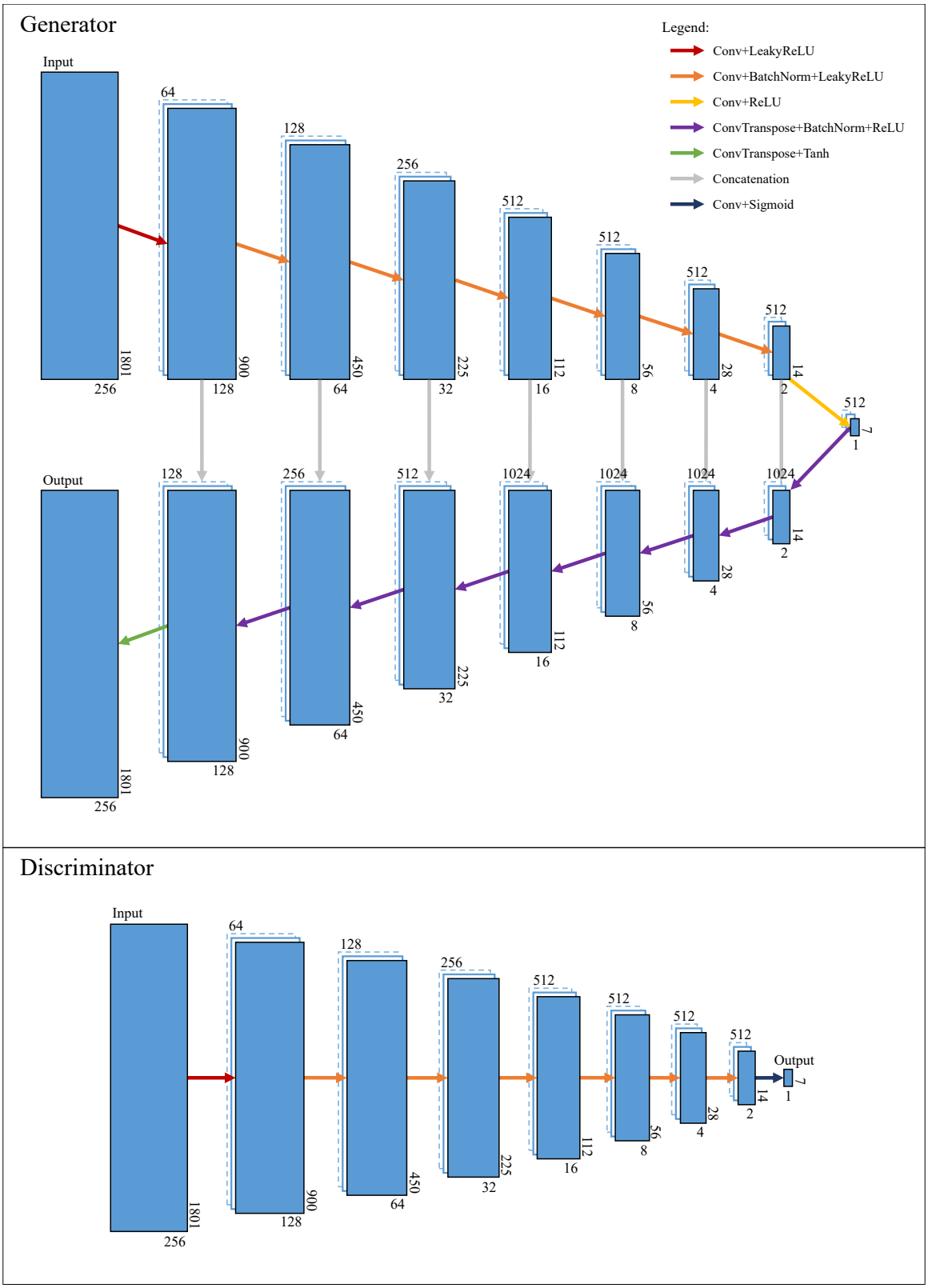

Fig. 7. The architecture of the Generator (top) and the Discriminator (bottom).

layer. The first four decoding layers have 1024 convolutional filters, after which the number of filters halves up to layer 7 with 128 filters, followed by the final decoding layer which has 1 filter, outputting a single-channel greyscale image.

Our discriminator architecture mirrors that of the generator's encoder (see the bottom image in Fig. 7). It has eight layers, each consisting of a 2D convolution with kernel size 4, stride 2 and padding 1, followed by batch normalisation with $\epsilon = 0.001$, and finally a leaky ReLU activation with slope 0.2. The first and final layers don't include batch normalisation, and the final layer's activation is a sigmoid

function. The number of convolutional filters for the discriminator begins at 64, and doubles every layer up to layer 4 with 512 filters. Layers 5, 6, and 7 have 512 filters, and the final layer has 1 filter, outputting a 7-dimensional vector. The final output probability of D is calculated as the mean of the components of this vector.

We optimise our models using Adam optimisation algorithm [27] with learning rate $\alpha = 0.0002$, $\beta_1 = 0.5$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$, and we set the weighting parameter $\lambda_{L^1} = 100$.

3. Data generation

3.1. Data collection

Measurements were collected at the Beamline I12-JEEP [28] at Diamond Light Source Ltd. A high-resolution pco.edge camera (Photon Lines Ltd.) with optical module 2 (pixel size 7.91 × 7.91 μm) was used for data collection with LuAG:Ce scintillator. The monochromatic X-ray radiation with energy $E = 55$ keV was selected. The detector resolution in pixels is 2160 × 2560.

The dynamic process of liquid flow (colloidal solution of CaCO3 in water) through highly porous sandstones (with random pore sizes from micron size up to approx. 2 mm in diameter) was used as a model experiment. Calcium carbonate was used as a contrast agent due to relatively high X-ray absorption. High-resolution tomographic scans of pure sandstones in transparent vials were performed before each dynamic experiment at different heights with vertical step 0.1 mm over a vertical translation of 2 mm. This means data can be recorded for multiple overlapping volumes with scintillator defects covering different features of the sandstone at different vertical translations.

A large amount of flats (200 frames) and darks (200 frames) were collected before each dynamic experiment, and during each dynamic experiment only projections were acquired. The tomography stage was continuously rotated at a fixed vertical translation for the duration of the dynamic experiment, allowing continuous acquisition of tomography data (1801 projections per tomography) with gaps. The calcium carbonate solution was slowly injected into the transparent vial with sandstones via a remotely controlled syringe pump (Harvard Apparatus) with 20 ml syringes. After the injection of the solution, the slow diffusion process of the contrast agent CaCO3 into the pores of sandstones was observed in reconstructed tomography data. The total length of each dynamic experiment exceeded 20 min, allowing the completion of the diffusion process.

In order to improve model generalisation, the datatset must include broader, more diverse samples. Two external datasets [36,37] were used: one was included in the training set [36], and another was used to analyse the model's performance on unseen data [37]. Two samples of the sandstone data that was used for training were uploaded to Zenodo [38].

3.2. Data generation

The training and testing datasets were created from five tomographic scans (see Sec. 3.1). To ensure successful training, pairs of artefact-free target images ('clean') and input images containing stripe artifacts ('stripe') were obtained (see Fig. 8). Each patch in a target-input pair must be identical apart from the simulated stripes, therefore the patches that contained stripes were excluded.

We generated our training dataset as follows: .

1. Applied the stripe detection algorithm (see Sec. 2.1) to the full volume of 1801 × 2160 × 2560 voxels to get a binary mask with the detected stripes.
2. Split each 1801 × 2560 sinogram into ten 1801 × 256 rectangular patches. Note that we preserved the full angular dimension (1801) in order to ensure the completeness of a stripe.
3. For each patch, retrieved the corresponding mask from the full mask volume. If the sum of the mask was $> = 1$, then the patch contained a stripe and was discarded.
4. Using only 'clean' patches, added simulated stripes of various width to the data using TomoPhantom software [29].

We also cropped out patches that did not contain any structural information or solely contained noise, as these are not useful for training and could affect the model's performance. Ultimately, we ended up with 75,487 target/input pairs, which were split into train, validate and test sets with ratio 3: 1: 1. This meant we had 45,293 training images, 15,097 validation images, and 15,097 testing images. We used a batch size of 16, giving 2831 training batches, 944 validation batches, and 944 testing batches.

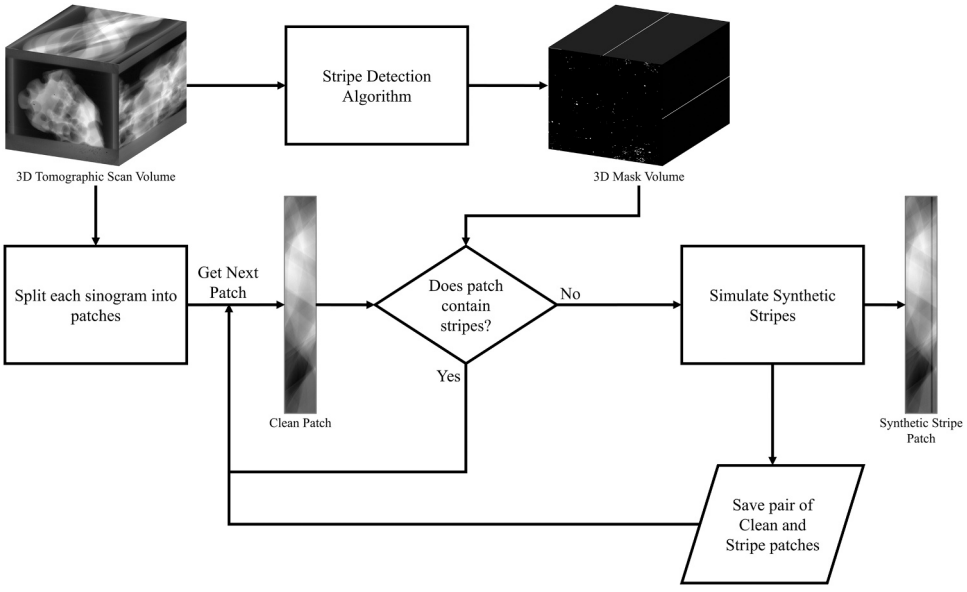

Fig. 8. This diagram shows the process of generating training data from real tomographic measurements for the subsequent use in the deep learning training. The stripe detection algorithm (see Sec. 2.1) helps to identify the 'clean' and 'stripe' data patches. Using the 'clean' data patches (no stripes present), the stripes are simulated and the pairs of targets and inputs are generated.

4. Results

4.1. Software

NoStripesNet[1] is a software package containing the Python code which reproduces results presented in the paper. It is written using the machine learning framework PyTorch [30] and consists of modules for data generation, pre-processing, model training and visualisation.

The model was trained for 100 epochs on 16 NVIDIA P100 GPUs, split across 4 nodes of a computer cluster. This parallelism was accomplished using PyTorch's *DistributedDataParallel* module, which in turn uses the NVIDIA Collective Communications Library. Training in this manner allowed us to achieve a speed of 4.9 times that of single-node, single-GPU training. The total training time was 18 h.

A graph of Generator and Discriminator losses can be seen in Fig. 9. The graph shows expected behaviour from the two networks; whenever Discriminator loss drops, Generator loss rises, reflecting the adversarial nature of GANs. Interestingly, the Discriminator was quite stable for the first 10 epochs as its loss did not vary much. This could be because, in the early epochs of training, the Discriminator had not yet learnt how to distinguish between real and fake images and mostly outputted a probability of 0.5. This would result in a binary cross entropy loss of 0.693, which is similar to that which is shown in Fig. 9. The model was then evaluated on the testing dataset, achieving a peak-signal-to-noise ratio (PSNR) of 45.114 for the whole region of interest (ROI). See more detailed analysis for separate ROIs in Tables 2 and 3. We also calculated the structural similarity metric [31], which resulted in values: 0.9851 for the whole ROI, 0.85 for the ROI with simulated stripes only and 0.91 for the ROI when stripes were inpainted using GANs.

We also trained our model for 100 more epochs, totalling 200, to assess whether more training would lead to better results. Fig. 10 shows the losses for the full 200 epochs.

One can see that training for more than 100 epochs led to worse model performance. Generator loss increases and Discriminator loss decreases rapidly, meaning the Discriminator is better able to tell the difference between real and fake images, which in turn means the Generator produces worse outputs. Therefore, we opted to use the model trained for 100 epochs when calculating any further results.

4.2. Stripes removal for real data

We measured the performance of the model in two ways. First, we applied the model to synthetic stripes on real-life data. This means we have a ground-truth 'clean' image with no stripes which can be used as a reference to perform a quantitative analysis. Second, we applied the model to real-life stripes. As there is no usable reference image in this case, only a qualitative analysis can be performed.

We also compared the model's performance to the algorithmic inpainting method described in Sec. 2, as well as two popular ring removal methods: Fourier Wavelet (FW) filtering [32], and algorithm no. 3 from [5] (Sorting).

4.2.1. Removing synthetic stripes

We used the simulated synthetic stripes from Sec. 3.2 to evaluate the ring removal methods. We created two volumes from this data; one that contains no artefacts ('clean') and another that contains synthetic stripes ('stripe'). We applied all methods to the 'stripe' volume, and then calculated the root mean squared error (RMSE) between the clean volume and the method's output. We calculated RMSE (see Table 2) in three different ways: on the whole sinogram, just stripe regions, and just non-stripe regions. Parameters for each method were optimised in a way to produce the lowest RMSE for the whole and the stripe region respectively.

As shown in Table 2, the algorithmic inpainting had the lowest RMSE and cGAN inpainting very close to it, meaning it was able to most accurately restore the artefact areas to their original values. Both inpainting methods had an RMSE of 0 in non-stripe regions, as they only change data in stripe regions. This reflects one of the main differences with ring removal filters where the whole data is modified. Generally the latter should be avoided or minimised, but with filters this is rarely possible (see more on this in Sec. 5).

The FW filtering method had the highest RMSE of all for whole sinograms and non-stripe ROIs. However, the sorting method had the highest RMSE for stripe ROIs. The FW method applies filtering in the Fourier space, after the Wavelet transform. It is possible that useful data components can be suppressed (filtered out) with the damping of vertical artefacts. The sorting method relies on the size of the median filter kernel for smoothing, which is related to the thickness of stripes present in the data. As the kernel of the filter is shift invariant, it is difficult to accommodate stripes of variable thickness with this algorithm (there is a separate algorithm for larger stripes in [5]).

We also performed a qualitative analysis of stripe removal performance for synthetic stripes, the results of which can be seen in Fig. 11. We show the output of each method for a particular stripe sinogram, as well as the residual between each output and the equivalent clean sinogram. As expected, the two filtering methods (FW and sorting) change both stripe and non-stripe ROIs, whereas the inpainting methods only change stripe ROIs. Interestingly, FW filtering appears to make lots of medium intensity changes to the image as a whole, as shown by the mostly uniform purple residual. This explains why its RMSE for stripe and non-stripe ROIs are within the same order of magnitude, unlike the other methods. The sorting method is able to smooth out and remove small, high frequency stripes, which none of the other methods are able to do. However, it also makes changes to non-stripe regions, as shown by the high intensity areas around the centre of the image.

In Table 3 we present a quantitative analysis of the reconstructed images with the simulated stripes using GridRec [33] reconstruction method from the TomoPy package [16]. Interestingly, in the reconstructed images applying the sorting algorithm resulted in the highest RMSE with the FW method close to it. Then the lowest value achieved using the algorithmic inpaiting and the cGAN method is slightly behind it. Although metric-wise the cGAN method is slightly underperforms compared to algorithmic inpainting, visually it is difficult to assess with certainty which one is better as we demonstrate it in the following section.

4.2.2. Removing real stripes

We use the normalised tomographic scan volume to analyse the performance of each method on real-life stripes. In order to know which sinograms contain real stripes, and so that the inpainting methods have masks to use, we use the 3D mask volume created using the stripe detection algorithm in Sec. 2.1.

Similar to synthetic stripes, in Fig. 12 we show the output of each method and its residual. There is no clean image to use as a reference, so the residual is instead calculated between the stripe image and the output of each method. This means that the residuals in this section represent how much each image was changed, rather than how close each image was to being artefact-free.

In Fig. 13, we also demonstrate the GridRec reconstruction applied to filtered and inpainted data from Fig. 12.

Similar to synthetic data, FW filtering seems to uniformly effect almost the entire image, and the reconstruction residual is also very high. This filtering causes additional artefacts, as the centre of the image is darker than the outer section. The sorting method effectively removes the stripes, as can be seen by the two bright vertical lines in the centre of the residual. However, it also has an effect on non-stripe data, as residue of the sinogram can be seen in the residual. In the reconstruction space, the sorting method has a more local effect, however there is still some non-stripe information being changed in the centre of the residual. The

[1] https://github.com/dkazanc/NoStripesNet

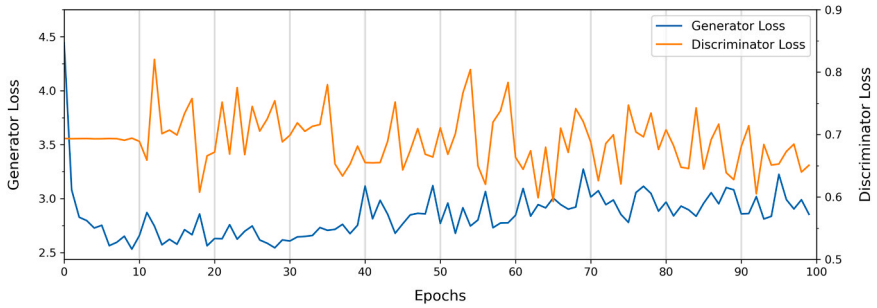

Fig. 9. Generator and Discriminator losses during training for 100 epochs. Generator loss is in blue with values on the left side of the graph, and Discriminator loss is in orange with values given on the right side of the graph. Generator peaks correspond to Discriminator troughs, and vice versa, which is expected due to the adversarial nature of GANs.

Table 2
Root mean squared Error (RMSE) on synthetic stripes of Fourier Wavelet (FW) filtering [32], algorithm no. 3 from [5] (Sorting), the algorithmic inpainting method described in Sec. 2.2, and the cGAN model. RMSE was calculated on the whole image (*whole*), just stripe regions (*stripe*) and just non-stripe regions (*non-stripe*). Notably the algorithmic inpainting provides the best result and the cGAN model inpainting is close to it in terms of RMSE. Both methods rely on the detection algorithm and therefore do not change the data outside the mask, hence RMSE in non-stripe ROI is zero for both.

Method	RMSE (whole)	RMSE (stripe)	RMSE (non-stripe)
Fourier Wavelet	2066.95	3013.01	2017.49
Sorting	677.88	3220.45	206.98
Alg. inpainting	**161.54**	**804.29**	0
cGAN Model	204.40	1017.64	0

Table 3
Root mean squared Error (RMSE) of the reconstructed volumes.

Method:	None	FW	Sorting	Alg. inpaint.	cGAN Model
RMSE	0.024	0.012	0.014	**0.0064**	0.0088

thicker rings are not fully removed with this method (see the magnified reconstructed images in Fig. 14).

The two inpainting methods successfully restore the data only within stripe regions, as can be seen in the reconstructions. Both of the methods suppress the rings effectively, but the algorithmic inpainting introduces some streaks into the reconstruction (more obvious in Fig. 14), whereas the cGAN inpainting does not. The cGAN reconstruction residual, however, does seem to have slightly larger residue values than algorithmic inpainting.

4.3. Model generalisation

We performed two tests to analyse how well the cGAN model is able to generalise to data unseen during its training.

One possible application of the proposed DL stripe removal method is in-situ dynamic tomographic imaging experiments [1]. Specifically, the studies of fluid dynamics processes in porous media [34,35]. The 'dry' tomographic scan for DL training can be initiated in the beginning of the experiment and the obtained model can then be used to process dynamic scans. In this case, the changes in the sample structure are not significant and we would expect the model to perform well.

Another test for model generalisation capabilities involved a totally new sample, with structure dissimilar to anything in the model's training dataset. In this case, it is unclear if the model will perform well.

4.3.1. Generalisation to dynamic data

In this case study, we applied the proposed method to a dynamic tomographic scan of a sandstone rock which had a solution of calcium carbonate dripped through it during the scan (see Sec. 3.1). Notably, the 'dry' scan of the same sample was included in the training data; we did not train on any of 'wet' data. The results are presented in Figs. 15 and 16.

Similar to Sec. 4.2, the two filtering methods change sample information in addition to artefact information. The sorting method is somewhat successful, suppressing, but not removing, the artefact entirely. The inpainting methods are both quite effective, and the algorithmic inpainting does not seem to introduce streaks here. In general, both inpainting methods seem to work quite well and the cGAN inpainting generalises well in this case of applying the model to unseen dynamic data.

4.3.2. Generalisation to unseen data

Here, we applied the cGAN model to unseen data sourced from [37], which is a totally different sample from those which are used in the training set. The scan was pre-processed using the dataset creation routine detailed in section 3.2. The other stripe removal methods used for comparison in section 4.2 were also applied to this dataset, and the results are shown in Figs. 17 and 18.

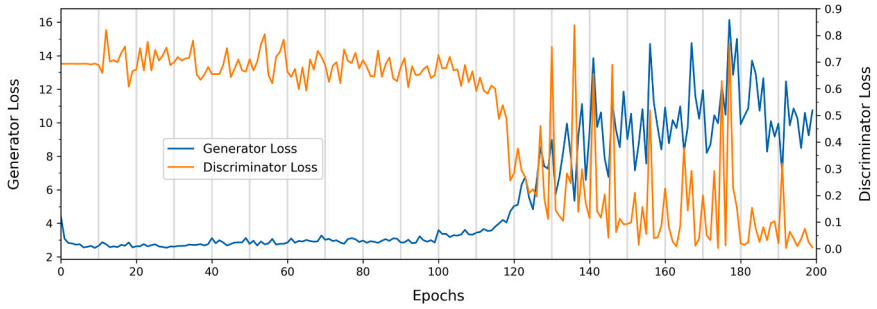

Fig. 10. Generator and Discriminator losses during training for 200 epochs. Generator loss is in blue with values on the left side of the graph, and Discriminator loss is in orange with values given on the right side of the graph. Notably, the performance worsens after 100 epochs, hence we used the model from the 100th epoch.

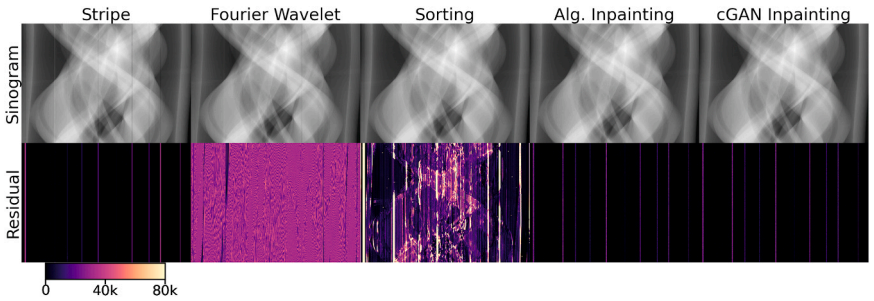

Fig. 11. Results of various methods on synthetic stripes. Top row shows sinograms, bottom row shows residuals between the output of each method and the ground truth clean sinogram.

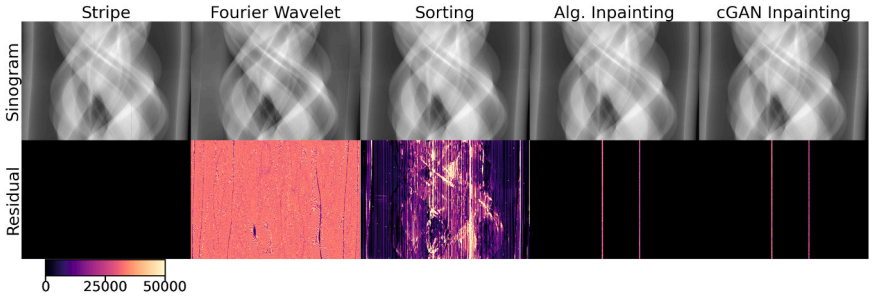

Fig. 12. Results of various methods on real stripes. Top row shows sinograms, bottom row shows residuals between the output of each method and the stripe sinogram.

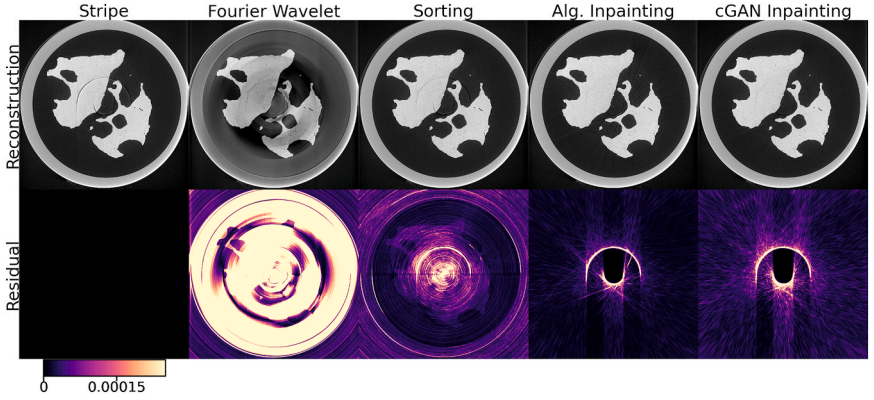

Fig. 13. Results of various methods on real stripes. Top row shows reconstructions, bottom row shows residuals between the output of each method and the stripe reconstruction.

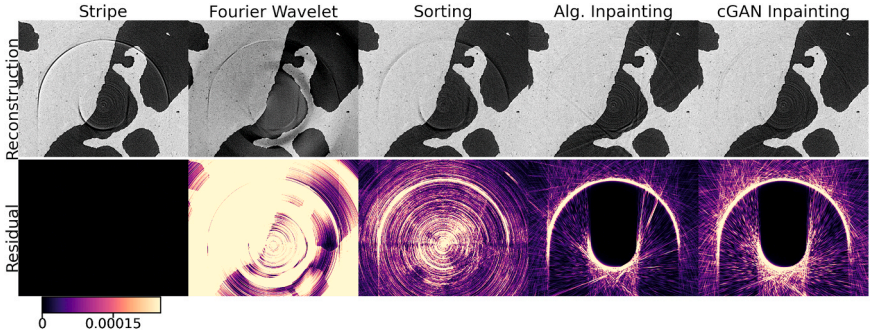

Fig. 14. Enlargement of ring artifact region in Fig. 13.

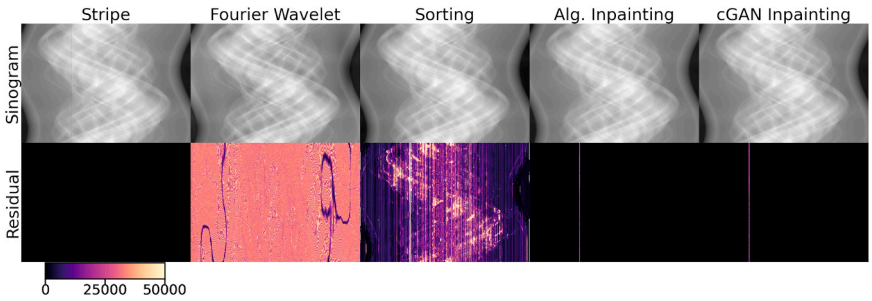

Fig. 15. Results of various stripe removal methods on a dynamic tomographic scan. The top row shows sinograms, and the bottom row shows the residual between each sinogram and the sinogram with stripes (top left).

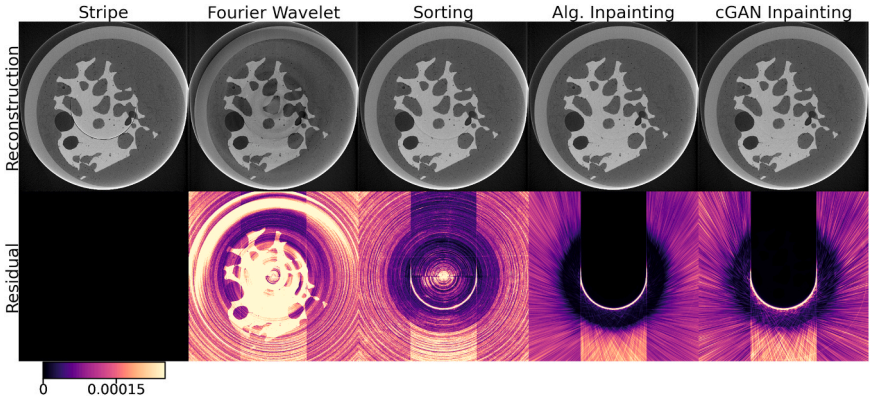

Fig. 16. Results of various stripe removal methods on a dynamic tomographic scan. The top row shows reconstructions, and the bottom row shows the residual between each reconstruction and the reconstruction with stripes (top left).

The model is quite successful at removing both stripes, however some residue of the larger stripe on the bottom can still be seen in Fig. 18. This residue is also present in the algorithmic inpainting, while both filtering methods fail to entirely remove the stripe. FW changes the whole image, which is shown especially well in the residual of the reconstruction. These residuals also show that the cGAN inpainting produces less streaks in reconstruction space than the algorithmic inpainting, even on data the model has not been trained on.

These tests show that the cGAN model is able to generalise quite well. It is able to successfully remove most artefacts and replace them with accurate, realistic data consistent with the rest of the sample, even when no similar samples exist in the training dataset. However, sometimes the model struggles to remove some artefacts, but the other methods can also be prone to errors (see Sec. 5).

5. Discussion and conclusions

In this paper, we presented three novel algorithms that were used to detect and suppress vertical stripe artefacts in sinograms. We used a novel stripe detection algorithm to locate the stripes in the data, which is a crucial step in identifying stripe-free regions which were used for deep learning training to inpaint the data. This approach makes it possible to re-use the collected raw data for training and apply the model to the same or similar unseen data. We demonstrated cases of successfully applying the network to dynamic in-situ data as well as to unseen data.

Arguably, data inpainting approach is not a generic solution for stripe or ring removal problems. There are cases when useful data in stripe regions can be restored almost exactly, without the need for new data generation. There are, however, situations when stripes in the data are severe ('unresponsive' or 'dead' stripes) and there is no any useful underlying information that can be used for a more gentle restoration. In this case, the presented data completion/inpainting methods would be the best choice. Furthermore, the detection of severe stripes could be a simple thresholding problem without the need of a sophisticated stripe detection algorithm.

Unfortunately, it is still problematic to come up with a generic stripe

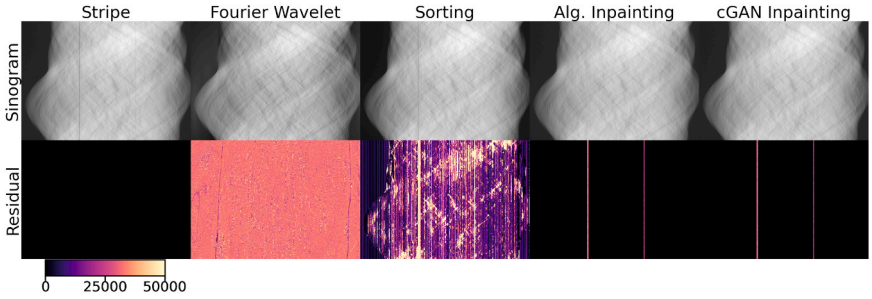

Fig. 17. Results of various stripe removal methods on tomographic data unseen by the cGAN model during training. The top row shows sinograms, and the bottom row shows the residual between each sinogram and the sinogram with stripes (top left).

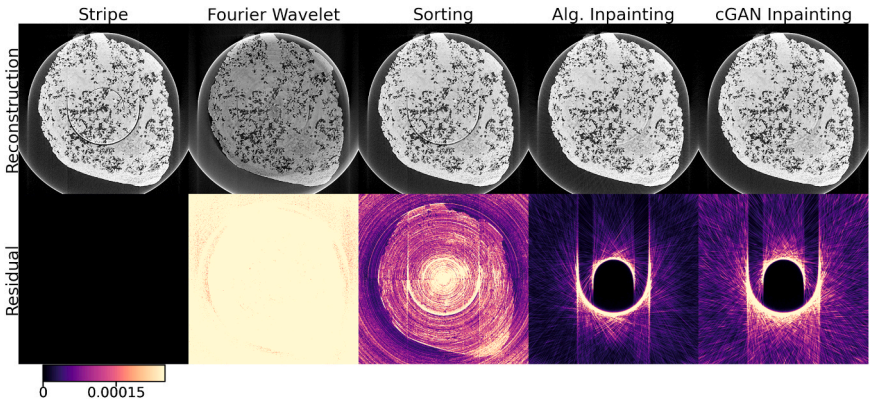

Fig. 18. Results of various stripe removal methods on tomographic data unseen by the cGAN model during training. The top row shows image reconstructions, and the bottom row shows the residual between each reconstruction and the reconstruction with stripes (top left).

removal method that would work equally well for all types of stripe artefacts. We are interested to explore the possibility of detecting and classifying different artefact types so that a particular stripe removal method can be applied. It is questionable if the whole process of stripes detection and restoration should be a part of one neural network. In this study, to make the problem less complex, we intentionally decoupled the general detection-removal problem so to specifically investigate the capabilities of the network to inpaint the data competitively.

In addition to the conditional generative adversarial network for inpainting, we presented a new algorithmic inpainting method, which delivers competitive results. Although the development of such algorithm was not the main goal of this paper, we demonstrated the superior performance of the method in terms of quantitative metrics compared to the deep learning method. Qualitatively, however, it is arguable which method performs better. The reconstruction using the algorithmically inpainted data, produces more streak artefacts than the deep learning method.

Possible improvements of the algorithmic inpainting could be more intelligent, rather than random, sampling. For example using Markov random field ideas of contextual connections in the images. This might have a better restorative effect on the existing boundaries in the data. Another interesting development will be to make the searching neighbourhood window in the method to be shift variant based on the distance from the region of usable data.

We would also like to add that the presented approach of collecting the training data using the available raw data and then applying the neural network, can be extended to other tomographic problems: removing metal artefacts or highly absorbing inclusions, removing outliers or clusters of outliers, and potentially other erroneous features in the data that can be detected and simulated.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

The authors acknowledge facilities and the support provided by Diamond Light Source. The authors acknowledge the Year in Industry scheme at Diamond Light Source that helped authors work on this project. The authors thank I12 beamline of DLS for the in-house beamtime provided (visit nt33730-1 in 2022) supporting this development.

References

[1] P.J. Withers, C. Bouman, S. Carmignato, V. Cnudde, D. Grimaldi, C.K. Hagen, E. Maire, M. Manley, A. DuPlessis, S.R. Stock, X-ray computed tomography, Nat. Rev. Methods Prim. 1 (1) (2021) 18, https://doi.org/10.1038/s43586-021-00015-.
[2] T.M. Buzug, Computed Tomography: From Photon Statistics to Modern Cone-Beam CT, Springer, 2008, https://doi.org/10.1007/978-3-540-39408-2.
[3] M. Bertero, P. Boccacci, C. De Mol, Introduction to Inverse Problems in Imaging, CRC Press, 2021, https://doi.org/10.1201/9781003032755.
[4] F.P. Vidal, J.M. Létang, G. Peix, P. Cloetens, Investigation ofartefact sources in synchrotron microtomography via virtual X-ray imaging, Nucl. Instrum. Methods Phys. Res. Sect. B Beam Interact. Mater. At. 234 (3) (2005) 333–348, https://doi.org/10.1016/j.nimb.2005.02.003.
[5] N.T. Vo, R.C. Atwood, M. Drakopoulos, Superior techniques foreliminating ring artifacts in X-ray micro-tomography, Opt. Express 26 (22) (2018) 28396–28412, https://doi.org/10.1364/OE.26.028396.
[6] K.A. Mohan, S.V. Venkatakrishnan, J.W. Gibbs, E.B. Gulsoy, X. Xiao, M. De Graef, P.W. Voorhees, C.A. Bouman, TIMBIR: a method for time-space reconstruction from interlaced views, IEEE Trans. Comput. Imaging 1 (2) (2015) 96–111, https://doi.org/10.1109/TCI.2015.2431913.
[7] H.O. Aggrawal, M.S. Andersen, S.D. Rose, E.Y. Sidky, A convexreconstruction model for X-ray tomographic imaging with uncertain flat-fields, IEEE Trans. Comput. Imaging 4 (1) (2017) 17–31, https://doi.org/10.1109/TCI.2017.2723246.
[8] D. Kazantsev, F. Bleichrodt, T. van Leeuwen, A. Kaestner, P.J. Withers, K. J. Batenburg, P.D. Lee, A novel tomographic reconstruction method based on the robust Studentas t function for suppressing data outliers, IEEE Trans. Comput. Imaging 3 (4) (2017) 682–693, https://doi.org/10.1109/TCI.2017.2694607.
[9] Yoo, S., Yang, X., Wolfman, M., Gursoy, D., Katsaggelos, A.K. Sinogram image completion for limited angle tomography with generative adversarial networks, in: Proceedings of the 2019 IEEE International Conference on Image Processing (ICIP), 2019, 1252–1256.10.1109/ICIP.2019.8804416.
[10] Li, Z., Zhang, W., Wang, L., Cai, A., Liang, N., Yan, B., Li, L. A sinogram inpainting method based on generative adversarial network for limited-angle computed tomography. In: Proceedings of the Fifteenth International Meeting on Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine, 11072, 2019, 345–349.10.1117/12.2533757.
[11] Z. Wang, J. Li, M. Enoh, Removing ring artifacts in CBCT images viagenerative adversarial networks with unidirectional relative total variation loss, Neural Comput. Appl. 31 (2019) 5147–5158, https://doi.org/10.1007/s00521-018-04007-6.
[12] M.U. Ghani, W.C. Karl, Fast enhanced CT metal artifact reduction using data domain deep learning, IEEE Trans. Comput. Imaging 27 (6) (2019) 181–193, https://doi.org/10.1109/TCI.2019.2937221.
[13] E. Valat, K. Farrahi, T. Blumensath, Sinogram inpainting withgenerative adversarial networks and shape priors, Tomography 9 (3) (2023) 1137–1152, https://doi.org/10.3390/tomography9030094.
[14] P. Isola, J.Y. Zhu, T. Zhou, A.A. Efros, Image-to-image translation with conditional adversarial networks, Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (2017) 1125–1134, https://doi.org/10.1109/CVPR.2017.632.
[15] E.M. Anas, S.Y. Lee, M.K. Hasan, Removal of ring artifacts in CTimaging through detection and correction of stripes in the sinogram, Phys. Med. Biol. 55 (22) (2010) 6911, https://doi.org/10.1088/0031-9155/55/22/020.
[16] D. Gürsoy, F. De Carlo, X. Xiao, C. Jacobsen, TomoPy: a framework for the analysis of synchrotron tomographic data, J. Synchrotron Radiat. 21 (5) (2014) 1188–1193, https://doi.org/10.1107/S1600577514013939.
[17] Bertalmio, M., Bertozzi, A.L., 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, I-I. IEEE. 10.1109/CVPR.2001.990497.

[18] C. Peng, B. Qiu, M. Li, Y. Guan, C. Zhang, Z. Wu, J. Zheng, Gaussian diffusion sinogram inpainting for X-ray CT metal artifact reduction, Biomed. Eng. Online 16 (1) (2017) 1–7, https://doi.org/10.1186/s12938-016-0292-9.
[19] H. Zhang, J.J. Sonke, Directional sinogram interpolation for sparseangular acquisition in cone-beam computed tomography, J. X Ray Sci. Technol. 21 (4) (2013) 481–496, https://doi.org/10.3233/xst-130401.
[20] Y. Li, Y. Chen, Y. Hu, A. Oukili, L. Luo, W. Chen, C. Toumoulin, Strategy of computed tomography sinogram inpainting based on sinusoid-like curve decomposition and eigenvector-guided interpolation, JOSA A 29 (1) (2012) 153–163, https://doi.org/10.1364/josaa.29.000153.
[21] A. Telea, An image inpainting technique based on the fast marching method, J. Graph. Tools 9 (1) (2004) 23–34, https://doi.org/10.1080/10867651.2004.10487596.
[22] Bertalmio, M., Sapiro, G., Caselles, V., Ballester, C. Image inpainting. In: Proceedings of the Twenty Seventh Annual Conference on Computer Graphics and Interactive Techniques, 2000, 417–424.10.1145/344779.344972.
[23] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, Adv. Neural Inf. Process. Syst. (2014) 27, https://doi.org/10.1145/3422622.
[24] Mirza, M., Osindero, S. , Conditional generative adversarial nets. arXiv Preprint arXiv:1411.1784.2014.10.48550/arXiv.1411.1784.
[25] Goodfellow, I. NIPS 2016 tutorial: Generative adversarial networks, arXiv preprint arXiv:1701.00160.2016.10.48550/arXiv.1701.00160.
[26] O. Ronneberger, P. Fischer, T. Brox, U-Net: convolutional networksfor biomedical image segmentation, Med. Image Comput. Comput. -Assist. Interv. (2015) 234–241, https://doi.org/10.1007/978-3-319-24574-4_28.
[27] Kingma, D.P., Ba, J. Adam: a method for stochastic optimization, in: Proceedings of the Third International Conference on Learning Representations, 2015.10.48550/arXiv.1412.6980.
[28] M. Drakopoulos, T. Connolley, C. Reinhard, R. Atwood, O. Magdysyuk, N. Vo, M. Hart, L. Connor, B. Humphreys, G. Howell, S. Davies, I12: The joint engineering, environment and processing (JEEP) beamline at diamond light source, J. Synchrotron Radiat. 22 (3) (2015) 828–838, https://doi.org/10.1107/s1600577515003513.
[29] D. Kazantsev, V. Pickalov, S. Nagella, E. Pasca, P.J. Withers, TomoPhantom, a software package to generate ^{2}D–^{4}D analytical phantoms for CT image reconstruction algorithm benchmarks, SoftwareX 7 (2018) 150–155, https://doi.org/10.1016/j.softx.2018.05.003.
[30] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, Pytorch: an imperative style, high-performance deep learning library, Adv. Neural Inf. Process. Syst. (2019) 32.
[31] Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Trans. Image Process. 13 (4) (2004) 600–612, https://doi.org/10.1109/TIP.2003.819861.
[32] B. Münch, P. Trtik, F. Marone, M. Stampanoni, Stripe and ringartifact removal with combined wavelet—Fourier filtering, Opt. Express 17 (10) (2009) 8567–8591, https://doi.org/10.1364/oe.17.008567.
[33] F. Marone, M. Stampanoni, Regridding reconstruction algorithm forreal-time tomographic imaging, J. Synchrotron Radiat. 19 (6) (2012) 1029–1037, https://doi.org/10.1107/s0909049512032864.
[34] K.J. Dobson, S.B. Coban, S.A. McDonald, J.N. Walsh, R.C. Atwood, P.J. Withers, 4-D imaging of sub-second dynamics in pore-scale processes using real-time synchrotron X-ray tomography, Solid Earth 7 (4) (2016) 1059–1073, https://doi.org/10.5194/se-7-1059-2016.
[35] A.P. Kaestner, P. Trtik, M. Zarebanadkouki, D. Kazantsev, M. Snehota, K.J. Dobson, E.H. Lehmann, Recent developments in neutron imaging with applications for porous media research, Solid Earth 7 (5) (2016) 1281–1292, https://doi.org/10.5194/se-7-1281-2016.
[36] [dataset] Vo, N.T., Atwood, R.C., Drakopoulos, M. Tomographic data for testing, demonstrating, and developing methods of removing ring artifacts, Zenodo, 2018.10.5281/zenodo.1443568.
[37] [dataset] Vo, N.T., Atwood, R.C., Drakopoulos, M. Tomographic data or demonstrating distortion correction methods, Zenodo, 2019.10.5281/zenodo.3339629.
[38] [dataset] Kazantsev, D., Magdysyuk, O., Beveridge, L. Sandstone rock tomographic data, i12 beamline, DLS synchrotron, Zenodo, 2023.10.5281/zenodo.10033401.