An Educated Warm Start For Deep Image Prior-Based Micro CT Reconstruction

Riccardo Barbano†, Johannes Leuschner†, Maximilian Schmidt, Alexander Denker, Andreas Hauptmann,
Peter Maass and Bangti Jin

***Abstract*—Deep image prior (DIP) was recently introduced as an effective unsupervised approach for image restoration tasks. DIP represents the image to be recovered as the output of a deep convolutional neural network, and learns the network's parameters such that the model output matches the corrupted observation. Despite its impressive reconstructive properties, the approach is slow when compared to supervisedly learned, or traditional reconstruction techniques. To address the computational challenge, we bestow DIP with a two-stage learning paradigm: (i) perform a supervised pretraining of the network on a simulated dataset; (ii) fine-tune the network's parameters to adapt to the target reconstruction task. We provide a thorough empirical analysis to shed insights into the impacts of pretraining in the context of image reconstruction. We showcase that pretraining considerably speeds up and stabilizes the subsequent reconstruction task from real-measured 2D and 3D micro computed tomography data of biological specimens. The code and additional experimental materials are available at educateddip.github.io/docs.educated_deep_image_prior/.**

I. Introduction

Inverse problems in imaging center around recovering an unknown image $x \in \mathbb{R}^n$ of interest from the noisy measurement $y_\delta = Ax + \eta$, where $y_\delta \in \mathbb{R}^m$ is the noisy measurement data, A the linear forward operator, and η an i.i.d. noise (e.g. Gaussian noise $\eta \sim \mathcal{N}(0, \sigma^2 I)$). Due to the inherent ill-posedness of the problem, suitable regularization is crucial and is key for a successful recovery of x [1]–[3].

Over the last years, deep learning methods have been successfully applied to solve all types of imaging problems, with supervised training being the dominant paradigm [4], [5]. That means, a deep neural network is trained to restore the image from noisy data using a set of paired training data. A large number of such high-quality paired training data may be needed [6]. Except simulated data, these are usually not obtainable, or too expensive to collect. Further challenges arise from the distributional shifts of the test data (e.g. change of image class, noise level or forward operator at test time). Ideally, the trained model should be robust to these changes, and transfer its reconstructive properties from one domain to another using as little additional data as possible [7]–[9]. Unfortunately, this is often not the case.

An effective solution to these challenges is deep image prior (DIP) [10], which represents a new approach to regularize image restoration. Rather than taking the supervised route, DIP learns to reconstruct without reference data, by assuming that a natural image can be well represented by a convolutional neural network (CNN). This is achieved by training the network's parameters to generate an image that fits the data y_δ (often equipped with suitable early stopping). The method is very attractive for imaging tasks with scarce training data. DIP has received enormous attention in the imaging community, and delivered state-of-the-art performance for unsupervised methods on a number of imaging tasks, including computed tomography (CT) [6], [11], magnetic resonance imaging (MRI) [12], positron emission tomography (PET) [13]–[15] and compressive ptychography [16], closely matching its supervised counterparts.

While DIP has been shown to be effective, it is not free from drawbacks. Notably, it requires "fresh training" each time it is deployed, which leads to high computational overhead and demanding VRAM requirements at test time when compared to supervised counterparts [4], [17], [18]; the latter ones only require one feed-forward pass through the network and thus computationally cheap. This inefficiency is considerably exacerbated by the fact that DIP requires a lengthy (and unstable) optimization process [6], [19]. For example, reconstructing a single image of resolution $(501\,\mathrm{px})^2$ requires approximately 30-50k iterations to reach the early-stopping point, which translates to 3-5 h of computing-time on NVIDIA GeForce RTX 2080Ti/1080Ti. It gets even worse in the 3D setting: a $(167\,\mathrm{px})^3$ reconstruction takes approximately one day on NVIDIA GeForce RTX 3090 using mixed precision! This hinders its applicability to solve imaging inverse problems, especially when fast reconstruction is critical. These observations motivate us to explore the following:

> *Can DIP benefit from pretraining for accelerating subsequent reconstructive tasks? If so, can we easily construct an informative dataset to warm-start DIP? How do inductive biases of pretraining impact the reconstructive task?*

Pretraining is one well-established paradigm to address data scarcity in supervised learning [20], [21]. Models are often pretrained using large-scale datasets, and fine-tuned on target tasks that have less training data [22]. However, the idea of pretraining has not received the attention it deserves for

R. Barbano is with the Department of Computer Science, University College London, UK (e-mail: riccardo.barbano.19@ucl.ac.uk).

J. Leuschner, M. Schmidt, A. Denker and P. Maass are with the Center for Industrial Mathematics, University of Bremen, Germany (e-mail: {jleuschn, maximilian.schmidt,adenker,pmaass}@uni-bremen.de).

A. Hauptmann is with the Research Unit of Mathematical Sciences, University of Oulu, Finland, and also with the Department of Computer Science, University College London, UK (e-mail: andreas.hauptmann@oulu.fi).

B. Jin is with Department of Mathematics, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong (e-mail: btjin@math.cuhk.edu.hk,bangti.jin@gmail.com)

† equal contributors.

DIP, and presents a new challenge. The challenge is to learn (via supervised pretraining) feature representations that are transferable and generalizable to subsequent fine-tuning.

To overcome the computational challenge, we systematically explore a supervised pretraining strategy for accelerating DIP-based μCT reconstruction, and introduce an effective two-stage learning paradigm. Our contributions can be summarized as follows. We develop an effective strategy to greatly accelerate the convergence of DIP for μCT reconstruction, by recasting DIP within the "supervised pretraining + unsupervised fine-tuning" paradigm. We show that carefully designed pretraining with simulated data from a synthetic image class can considerably speed up and stabilize DIP-based μCT reconstruction with real-measured data, including computationally demanding 3D tasks, for which we develop a specialized U-Net architecture to perform DIP-based μCT reconstruction under the constraint of 24 GB VRAM. To the best of our knowledge, this is the first successful 3D μCT reconstruction using DIP. The experiment results show that despite its simplicity, it can be highly effective. Further, we conduct a thorough experimental study to shed insights into the mechanism of knowledge transfer between the supervised pretraining and unsupervised fine-tuning stages, including a novel linear analysis of pretraining, which exhibits sparsity-promoting in the parameters' bases.

The paper is organized as follows. We describe the standard DIP in Section II and related works in Section III. In Section IV, we present the two-stage framework for DIP. We give the experimental details and results in Sections V and VI, and analyze the impact of pretraining in Section VII.

II. Deep Image Prior

The idea of DIP [10] is to find a minimizer of the fidelity $\|Ax - y_\delta\|^2$, by representing the unknown x as the output of a CNN, $x = \varphi_\theta(z)$, where $z \in \mathbb{R}^n$ is a fixed random vector (often pixel-wise i.i.d. samples of random noise), and $\theta \in \mathbb{R}^p$ denotes the network's parameters to be learned. A U-Net [23] like architecture is commonly used for the network. DIP solves

$$\theta^* \in \underset{\theta}{\operatorname{argmin}} \|A\varphi_\theta(z) - y_\delta\|^2,$$

and presents $\varphi_{\theta^*}(z)$ as the reconstruction. Note that the training of the network parameters θ coincides with the recovery process, and has to be repeated for each measurement. The procedure is unsupervised, and guided by the principle of matching the forward projected network output $A\varphi_\theta(z)$ to the measurement data y_δ. Due to the overparameterization of the neural networks used in DIP, a direct minimization of the loss can suffer from overfitting. DIP often uses early-stopping to deliver a satisfactory reconstruction: the update of θ is stopped early to avoid overfitting to the noise [10]. This has motivated developing automated rules for early stopping [24], [25].

III. Related Works

a) Deep Image Prior: Since the first proposal in [10], there have been several important developments on DIP. Heckel et al. [26] propose deep decoder, using under-parameterized networks to ease the need for early-stopping. Dittmer et al. [27] study DIP through the lens of regularization theory [1]–[3], and Cheng et al. [28] discuss its connection with Gaussian processes as the number of architecture channels grows to infinity, and propose the use of Bayesian learning. There are several efforts to combine DIP with explicit regularization to improve the reconstruction quality. [6], [29] propose the use of total variation penalty for stabilizing the learning process, and [30] combines DIP with regularization by denoising. Besides, the use of explicit regularization significantly relaxes the need of early stopping. Jo et al. [24] propose to penalize the complexity of the reconstruction using Stein's unbiased risk estimator. See also [25] for a stopping criterion based on monitoring the running variance of iterate sequence and references therein for further discussions. [6] suggests at test-time to start optimizing a randomly initialized DIP to match a reconstruction, produced by another method. Heckel and Soltanolkotabi [31] prove that for compressed sensing, an untrained CNN can approximately reconstruct signals and images that are sufficiently structured, from a near minimal number of random measurements. The very recent work [32] establishes the equivalence of "analytic" DIP with the standard Tikhonov regularization, and several basic properties in the lens of classical regularization theory. This work complements and expands on these existing studies by addressing the computational challenge associated with regularized DIP, especially the works [6], [29], where the regularized DIP was proposed and empirically demonstrated.

b) Advances in Pretraining: Supervised pretraining on ImageNet has been established as a common practice in computer vision. Neural networks are pretrained to solve image classification, and transferred to downstream tasks (e.g. object detection [33], [34] and semantic segmentation [35]). However, pretraining on ImageNet does not necessarily improve the accuracy of the downstream task [36], and similar observations about pretraining on ImageNet are made about medical image classification [37]. Within tomographic imaging, several works [38], [39] employ transfer learning to adapt a trained neural network from one task setting to another. Our work shares similarities with these works in adapting to changes of the image distribution. These works focus on supervised end-to-end fine-tuning, whereas we focus on an unsupervised learning framework: we study pretraining with a synthetic dataset as a means for accelerating DIP reconstruction on measured μCT data, and provide a detailed analysis of its acceleration mechanism. Very recently, Gilton et al [7] proposes to fine-tune the pretrained model so as to accommodate model errors, but unlike this work, the image distribution is unchanged. Inspired by an early version of this paper, Knopp and Grosser [11] also demonstrated the potential of warm-starting DIP for dynamic tomography.

IV. Proposed Method

The TV-regularized DIP approach obtains x^* by

$$\theta_{\mathrm{t}}^* \in \underset{\theta}{\operatorname{argmin}} \Big\{ l_{\mathrm{t}}(\theta) := \|A\,\varphi_\theta(z) - y_\delta\|^2 + \gamma\,\mathrm{TV}\,(\varphi_\theta(z)) \Big\},$$
$$x^* = \varphi_{\theta_{\mathrm{t}}^*}(z), \tag{1}$$

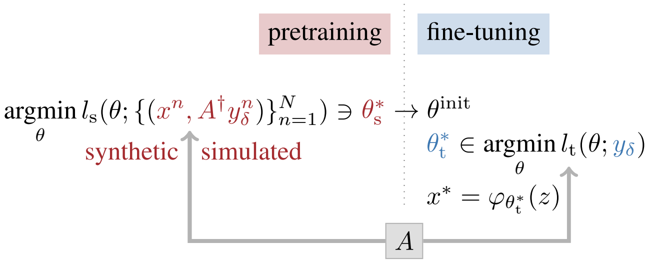

pretraining
fine-tuning
$\operatorname{argmin}_\theta l_{\rm s}(\theta; \{(x^n, A^\dagger y_\delta^n)\}_{n=1}^N) \ni \theta_{\rm s}^* \rightarrow \theta^{\rm init}$
synthetic
simulated
$\theta_{\rm t}^* \in \operatorname{argmin}_\theta l_{\rm t}(\theta; y_\delta)$
$x^* = \varphi_{\theta_{\rm t}^*}(z)$
A

Fig. 1. A two-stage learning paradigm. The parameters θ of the U-Net are first optimized on a dataset comprising ordered pairs of synthetic ground truth images x^n and simulated measurement y_δ^n. The optimal configuration $\theta_{\rm s}^*$ is then used to warm-start the unsupervised fine-tuning on real μCT data.

where φ_θ is a CNN, and $\gamma \geq 0$ balances the data consistency with the regularization term $\mathrm{TV}(\varphi_\theta(z))$, which denotes the total variation seminorm on the network output $\varphi_\theta(z)$, defined as $\mathrm{TV}(x) = \|\nabla_h x\|_1 + \|\nabla_v x\|_1$, where ∇_h and ∇_v denote the derivative in the horizontal and vertical directions. Several studies [6], [29] found that incorporating the total variation penalty is beneficial to DIP. The loss $l_{\rm t}$ in (1) is optimized with Adam [40], by randomly initializing θ. The learning is performed as (single-batch) test time adaptation to y_δ.

In this work, we recast DIP into the "supervised pretraining + unsupervised fine-tuning" paradigm as a two-stage process, called educated DIP (EDIP); see Fig. 1 for a schematic illustration of the framework. In the first stage, we pretrain the network $\varphi_\theta(A^\dagger y_\delta)$, where $A^\dagger$ is an approximate inverse operator (e.g. filtered back-projection (FBP) for CT [41]). The training is carried out on a synthetic dataset $\mathcal{D} = \{(x^n, y_\delta^n)\}_{n=1}^N$, composed of N pairs drawn from the joint distribution of ground truth x^n and corresponding simulated measurement y_δ^n. This step is tailored to the target reconstruction task in (1), and learns the optimal parameters $\theta_{\rm s}^*$ via supervised training,

$$\theta_{\rm s}^* \in \operatorname*{argmin}_\theta \Big\{ l_{\rm s}(\theta) := \frac{1}{N} \sum_{(x^n, y_\delta^n) \in \mathcal{D}} \|\varphi_\theta(A^\dagger y_\delta^n) - x^n\|^2 \Big\}. \quad (2)$$

Note that φ_θ receives $A^\dagger y_\delta^n$ as its input (instead of the random noise in [10]), serving as a post-processing reconstructor [42]. The objective of this stage is to enforce "benignant" inductive biases via supervised learning. This educates DIP with knowledge contained in the dataset $\mathcal{D}$, which is then exploited, but still needs to be amended, in solving the reconstruction task in (1).

In the second stage, for a given new query measurement y_δ, we use the optimal parameters $\theta_{\rm s}^*$ obtained in the pretraining stage to initialize the network $\varphi_\theta(A^\dagger y_\delta)$ in (1) so as to get DIP up to speed in handling target tasks on real-measured data. That is, we regard the DIP optimization as a self-adaptation step, where the parameters θ are fine-tuned unsupervisedly, with their drift conditioned on $\theta_{\rm s}^*$. Note that the robustness of this method at test time does not rely solely on how well the pretraining stage anticipates distributional shifts. The model makes a good use of pretraining — the supervised pretraining stage sets and constrains the stage — but adapts to distributional shifts at test time, and reserves its right to amend the received supervision.

There are several possible variants of the basic framework. U-Net consists of two parts, a decoder with parameters $\theta_{\rm dec}$, and an encoder with parameters $\theta_{\rm enc}$. A direct variant of EDIP is to fine-tune only the decoder parameters $\theta_{\rm dec}$, but fixing the encoder parameters to the educated guess $\theta_{\rm s,enc}^*$, which are regarded as a shared (between stages) feature extractor. At test time, we solve (1) only with respect to $\theta_{\rm dec}$ and rely on the pretraining to construct a suitable "universal" encoding. Thus, the learned reconstructor φ_{θ^*} recovers from the measurement data with $\theta^* = (\theta_{\rm s,enc}^*, \theta_{\rm t,dec}^*)$. This variant with the fixed encoder (FE) is termed as EDIP-FE.

V. Datasets

A. Synthetic Training Dataset

We pretrain on a synthetic training dataset of images composed of ellipses or ellipsoids with random position, shape, orientation and intensity values, which are commonly used to train and evaluate learned reconstruction methods. This image class encompasses basic building blocks of more complex images, while favoring piece-wise smoothness. Synthetic data is particularly useful when it is infeasible to collect high-quality ground truth images reassembling the image class of the target reconstructive task, while enabling the learning of features tailored to the inversion of the forward operator A. In the experiments, we use datasets of 32 000 training and 3200 validation images generated on-the-fly using ODL [43]. The image resolution and the distribution of the ellipses / ellipsoids can be easily adapted to match different target data. The synthetic projection data is computed by forward projecting the ground truth images and adding 5 % white noise. Fig. 2 shows an exemplary ground truth image and reconstructions obtained by the FBP and U-Net from the simulated noisy data.

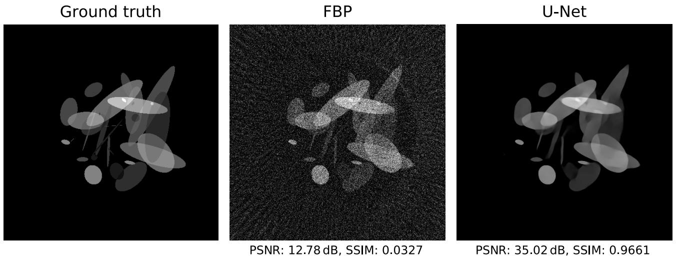


Fig. 2. An exemplary ground truth image used in the pretraining stage. The FBP and U-Net reconstructions are also shown. The measurement data y_δ is simulated using the Walnut `Sparse 120` setting, adding 5% white noise.

B. Real μCT Measurement Data

We evaluate our approach on two real μCT datasets to showcase the effectiveness of the approach. The forward operator A is a ray transform matching a 2D or 3D cone-beam geometry (cf. Fig. 3 for 2D). The scanner rotates around the object (or, equivalently, the object is rotated inside the scanner), taking projections from different source angles λ. Within each projection, each detector pixel (e.g. parameterized by γ) measures the intensity for a specific line, attenuated by the object.

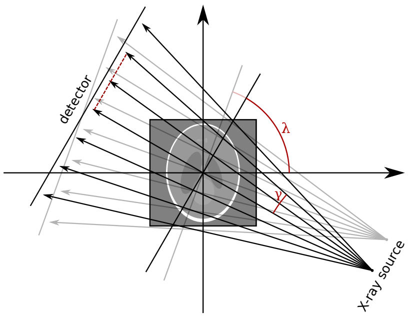


Fig. 3. Diagram of the 2D cone-beam geometry (a.k.a. fan-beam geometry).

a) X-ray Lotus Root Dataset: μCT measurements of a Lotus root slice filled with different materials are available from [44]. The dataset contains fan-beam measurements corresponding to a 2D volume slice, with 120 projections at angles equally distributed over $[0, 360°)$ and 429 detector pixel values each. The sparse matrix modeling the forward operator for an image resolution $(128\,\mathrm{px})^2$ is used. In the evaluation, we consider the setting of `Sparse 20`: a 6-fold angular subsampling, 20 angles, equally distributed over $[0, 360°)$. We use a TV-regularized reconstruction from all 120 projection angles, obtained by Adam, as the reference solution.

b) X-ray Walnut Dataset: A collection of cone-beam μCT measurement data from 42 Walnuts was provided in [45]. For each walnut, a set of three 3D cone-beam measurements is included, each obtained with a different source position. Projections are acquired at 1200 angles equally distributed over $[0, 360°)$, with a resolution of 972 detector rows and 768 detector columns. A volume resolution of $(501\,\mathrm{px})^3$ is used. We consider reconstructing a single 2D slice from a suitable subset of detector pixel measurements, and 3D reconstruction with a downscaled image resolution of $(167\,\mathrm{px})^3$. For the 2D task, we use the setting of `Sparse 120`: a 10-fold angular sub-sampling with 120 angles, equally distributed over $[0, 360°)$; for 3D, we consider the settings `3D Sparse 20` and `3D Sparse 60` with 20 and 60 equally distributed angles, and sub-sample the projection rows and columns by a factor of 3. The 3D settings are chosen to mimic industrial applications, where a high degree of sparsity is often desired. The approximations $A^\dagger y_\delta$ are computed via the Feldkamp-Davis-Kress (FDK) algorithm [46]. FDK is an FBP-based algorithm with a weighting step for cone-beam measurements, and is still denoted as “FBP”. To achieve accurate automatic differentiation of the forward projection operator in 2D, we utilize its sparse matrix representation. In 3D, we opt for forward and backward projection routines of ASTRA via tomosipo [47]. We use the ground truth provided with the dataset [45], which was obtained with accelerated gradient descent using the measurements from all 1200 projection angles and all three source positions.

VI. Experiments and Results

Throughout, we denote the type of the network input z used for a method in brackets: for example, “DIP (noise)” refers to the standard DIP with noise input, while “EDIP (FBP)” stands for the educated DIP with FBP input.

A. Neural Network Architecture

For 2D settings, we adopt the U-Net proposed by [6], but replace batch-normalization layers with group-normalization layers. For 3D μCT reconstructions, we fine-tune the architecture, cf. Fig. 4, since the standard 3D U-Net — originally introduced for segmentation [48] — does not meet our memory constraint, and a naively reduced version leads to sub-optimal reconstructions. We modify the U-Net architecture as follows: (i) reduce the numbers of channels per convolutional layer in the encoder; (ii) increase the expressivity of the decoder by chaining subsequent convolutional layers with decreasing number of channels; (iii) remove skip connections. Due to memory constraints (i.e. 24 GB VRAM), we use a 3-scale 3D U-Net.

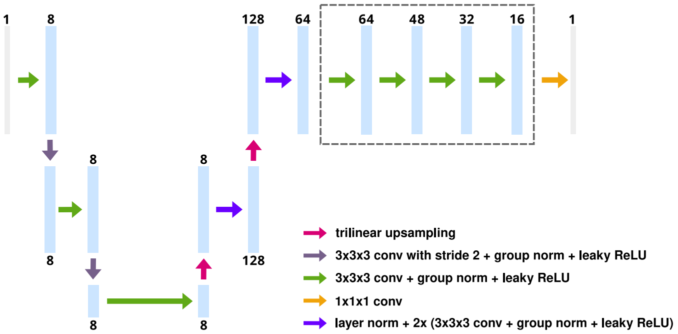

1
8
128
64
64
48
32
16
1
8
8
8
128
8
8
trilinear upsampling
3x3x3 conv with stride 2 + group norm + leaky ReLU
3x3x3 conv + group norm + leaky ReLU
1x1x1 conv
layer norm + 2x (3x3x3 conv + group norm + leaky ReLU)

Fig. 4. The architecture of the proposed 3D U-Net. Each light-blue bar corresponds to a multi-channel feature map. Arrows denote the different operations.

B. Evaluation Metrics

We measure the reconstruction quality via peak signal-to-noise ratio (PSNR), and include structural similarity index measure (SSIM) [49] for reconstructions. To assess the convergence speed, we employ two metrics: steady PSNR and rise time (denoted by $\star$ in the figures). The steady PSNR is the median PSNR over the last 5k iterations. The rise time is the iteration number at which we reach the baseline PSNR (i.e. DIP’s steady PSNR) up to a threshold $0.1\,\mathrm{dB}$. In addition, we always consider the iteration-wise median PSNR over repeated runs of the same experiment (with varying seeds) for these metrics; we use 5 runs for 2D and 3 runs for 3D. The variability between runs does arise not only from random initialization of the network parameters or noise input, but also from numerical effects in parallel computations on GPU. The optimal reconstruction $\varphi_{\theta_{\text{min-loss}}}(z)$ is taken from the iteration with minimum loss value $l_\mathrm{t}(\theta_{\text{min-loss}}) = \min_{i\in 0\ldots N} l_\mathrm{t}(\theta^{[i]})$. This remedies non-monotonous loss minimization, yet the (E)DIP optimization plots and steady PSNR computations use the actual iterate to facilitate a direct analysis.

C. Hyperparameter’s Selection

The learning rate and regularization parameter γ are fine-tuned for standard DIP. For EDIP, we do not conduct additional hyperparameter search, but use the values identified for DIP. These hyperparameters values also perform well for EDIP,

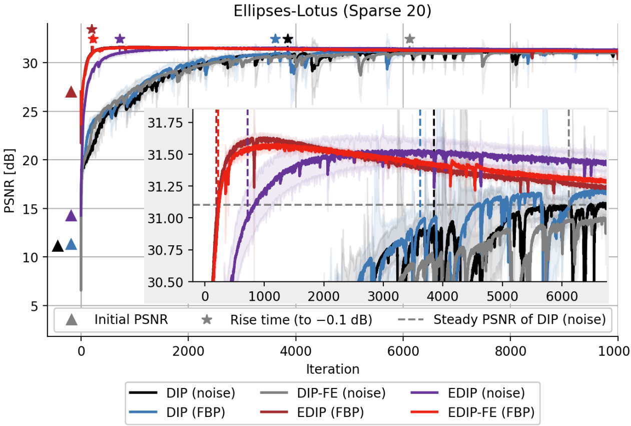


Fig. 5. The optimization of EDIP versus DIP on Lotus `Sparse 20`. The symbols ⋆ and ▲ denote initial PSNR and rise time, respectively, and the horizontal dashed line indicates the steady PSNR of DIP (noise).

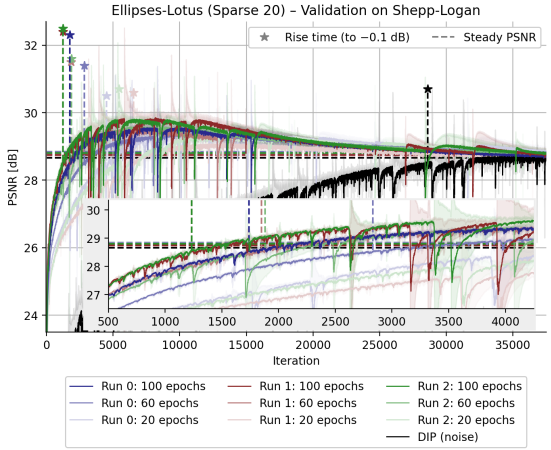


Fig. 6. Checkpoint selection on the Shepp-Logan phantom for the initial EDIP (FBP) model parameters for the Lotus `Sparse 20` setting.

which saves us from performing an individual search for each pretraining checkpoint (to be defined next).

D. Selection of the Checkpoints

Multiple parameters' configurations (i.e. θ^*_{s}) may be obtained from the pretraining stage: repeated runs (varying random initializations) and multiple checkpoints along the optimization trajectory of each run. From a set of checkpoints, one needs to identify solutions maximizing the speed-up at test time. More broadly, this is an open question. In the experiments below, the selection strategy is based on assessing the performance on the Shepp-Logan phantom [50], a standard test image within the medical imaging community. The checkpoint leading to the shortest rise time is then selected, among those with a steady PSNR that is at most 0.25 dB lower than the maximum steady PSNR of any checkpoint. This selection is carried out for 2D reconstruction settings; 3D runs use the best performing checkpoint for computational reasons. For Lotus `Sparse 20`, we repeat the pretraining 3 times (varying the seed) and collect checkpoints after every 20 epochs, training for a maximum of 100 epochs. For the Walnut `Sparse 120`, we pretrain for 20 epochs, and retain the minimum validation loss checkpoint of each run. For the 3D Walnut settings, we pretrain for a maximum of 2 epochs, and retain checkpoints every 0.125 epochs (i.e. 4k gradient updates).

TABLE I
QUANTITATIVE EVALUATION FOR THE LOTUS `SPARSE 20`.

Ellipses-Lotus `Sparse 20`

	Rise time	(Max PSNR; iters)	Steady PSNR
DIP (noise)	3848	(31.17; 8846)	31.10
DIP (FBP)	3622	(31.25; 8813)	31.17
DIP-FE (noise)	6118	(31.10; 9818)	31.00
EDIP (FBP)	195	(31.65; 981)	31.21
EDIP (noise)	723	(31.53; 3548)	31.39
EDIP-FE (FBP)	226	(31.59; 1421)	31.26
TV	–	–	30.73

E. The Lotus Root

Table I shows the convergence properties of EDIP and DIP for Lotus `Sparse 20`. We include in our analysis cases where the FBP $A^\dagger y_\delta$ is fed as the input (instead of noise) when solving (1) for DIP, and inputting noise for EDIP. EDIP significantly outperforms DIP in terms of the convergence speed for either a fixed noise image or FBP.

EDIP only takes 195 (and 723 for noise input) iterations to reach -0.1 dB of the baseline PSNR, against 4.1k iterations needed for DIP. Thus, pretraining greatly accelerates the convergence. The optimization process is considerably more stable (cf. Fig. 5), implying a possibly much more favorable loss landscape for EDIP. Thus, pretraining stabilizes the optimization process of DIP, which is highly desirable in practice. Note that EDIP-FE, which fixes the encoder parameters to the pretrained ones $\theta^*_{s,\mathrm{enc}}$, is as fast as EDIP, and the reconstruction quality of EDIP and EDIP-FE are largely comparable with each other. With fewer parameters to be updated, EDIP-FE is computationally lighter than EDIP (since backpropagation is only needed for the decoder, and the forward pass through the encoder can be pre-computed beforehand). Fig. 7 shows the reconstruction (along with the reference and FBP) for Lotus `Sparse 20`. We observe that pretraining can also boost the performance of DIP: EDIP considerably overshoots the baseline PSNR, cf. Fig. 5. This suggests that pretraining, if coupled with proper early-stopping (approximately a few hundred iterations after the rise time),

TABLE II
CHECKPOINTS' COMPARISON FROM THE PRETRAINING STAGE FOR EDIP (FBP) ON LOTUS `SPARSE 20`. THE CHECKPOINT FROM RUN 2 AFTER 100 EPOCHS IS SELECTED USING THE SHEPP-LOGAN DATA (CF. FIG. 6)

	Epochs	Rise time	(Max PSNR; iters)
Run 0	100	247	(31.49; 1545)
	60	174	(31.56; 842)
	20	291	(31.61; 1614)
Run 1	100	162	(31.53; 779)
	60	243	(31.53; 1755)
	20	390	(31.56; 1518)
Run 2	100	195	(31.65; 981)
	60	194	(31.58; 1083)
	20	318	(31.51; 1706)

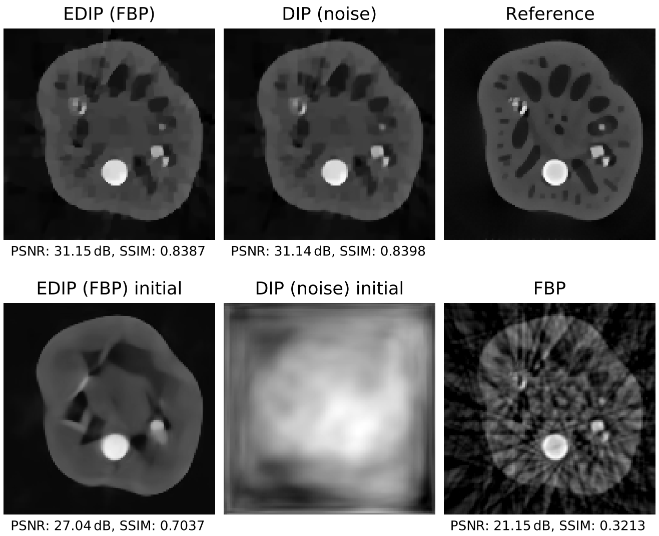

EDIP (FBP)
DIP (noise)
Reference
PSNR: 31.15 dB, SSIM: 0.8387
PSNR: 31.14 dB, SSIM: 0.8398
EDIP (FBP) initial
DIP (noise) initial
FBP
PSNR: 27.04 dB, SSIM: 0.7037
PSNR: 21.15 dB, SSIM: 0.3213

Fig. 7. EDIP versus DIP reconstruction on Lotus `Sparse 20`. From the 5 runs (varying the seed), the one with the (closest to) median PSNR was selected for each method.

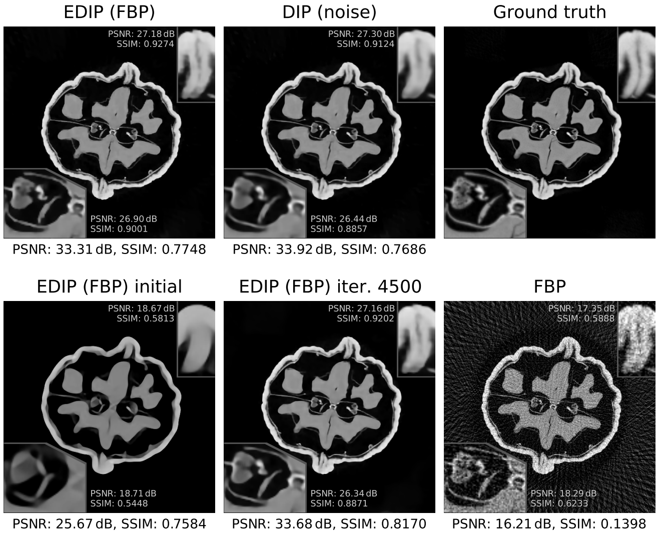


Fig. 8. EDIP versus DIP reconstruction of Walnut `sparse 120`.

TABLE III
QUANTITATIVE EVALUATION FOR THE WALNUT.

Ellipses/Ellipsoids-Walnut	`Sparse 120`			`3D Sparse 20`			`3D Sparse 60`		
	Rise time	(Max PSNR; iters)	Steady PSNR	Rise time	(Max PSNR; iters)	Steady PSNR	Rise time	(Max PSNR; iters)	Steady PSNR
DIP (noise)	20 373	(34.02; 25 357)	33.87	17 200	(30.68; 23 477)	30.37	49 041	(34.05; 58 901)	33.93
DIP (FBP)	13 778	(34.07; 28 094)	33.90	13 016	(31.32; 25 063)	31.19	27 873	(34.37; 53 731)	34.22
EDIP (FBP)	4496	(33.92; 13 039)	33.56	3739	(31.48; 10 689)	30.94	11 247	(34.35; 40 810)	34.18
EDIP-FE (FBP)	4384	(33.91; 12 540)	33.70	2979	(31.38; 10 749)	30.93	14 520	(34.33; 45 259)	34.15
TV	–	–	31.67	–	–	28.89	–	–	33.35

can lead to better reconstructions.

To maximize the speed-up, we select the warm-start configuration θ_{s}^* on the Shepp-Logan phantom. Fig. 6 shows the validation runs. We select θ_{s}^* from run 2 after 100 epochs since it results in the smallest rise time. Interestingly, a substantial overshoot of baseline PSNR is observed on the Shepp-Logan phantom, possibly due to its in-distribution nature with respect to the ellipses. Table II reports the rise time at test time for different checkpoints collected for each run. The results indicate that the checkpoint selection does impact the achievable acceleration factor, but not the maximum PSNR.

F. The Walnut

Fig. 8 shows the reconstructed Walnut slice; see Table III for quantitative results. A speed-up is observed, similar to the Lotus root: EDIP takes about 30 min at rise time (approximately 4.4k iterations), whereas DIP (with noise input) takes 2 h and 30 min at rise time (approximately 20.4k iterations) with NVIDIA GeForce RTX 2080Ti. A TV regularized reconstruction of the Walnut takes 6 min, and requires 1.7k gradient steps to converge to 31.67 dB. EDIP takes only 3 min (after 421 iterations) to match 31.67 dB. In 6 min, EDIP reaches 32.80 dB, with a gain of 1.1 dB. Finally, DIP-FE / EDIP-FE report similar performances to DIP / EDIP.

On a minor note, it is observed that EDIP better reconstructs finer structures (e.g. the wrinkled shell), and DIP suffers from over-smoothing artifacts. This concurs with the observation for Lotus `Sparse 20`: by incorporating the knowledge contained in the synthetic training data, pretraining can boost the performance of DIP.

Similar observations can be made for reconstructing the Walnut volume, cf. Fig. 9 for 3D reconstructions along the yz, xz, and xy axes and Table III for quantitative results. EDIP reconstruction from the `3D Sparse 20` data takes approx. 1.5 h with a NVIDIA GeForce RTX 3090, and leads to 33.77 dB in PSNR, compared to 7.3 h and 5.53 h for DIP (with noise / FBP as input). EDIP matches the PSNR of a TV reconstruction in about 30 min, gains 1 dB over TV after additional 20 min, and it takes 2.3 h to observe a 2 dB gain. The `3D Sparse 60` leads to similar speed-up. It takes 20 h for DIP with noise as input. Inputting the FBP results already in a considerable speed-up (about 11 h), whereas EDIP requires only 4 h. In sum, pretraining on the synthetic ellipsoids dataset greatly accelerates the convergence of DIP for 3D μCT reconstruction.

Last, we briefly comment on the convergence of the optimization process, cf. Fig. 10. The overall convergence behavior for 2D and 3D is similar to Lotus `Sparse 20`: pretraining stabilizes DIP optimization and greatly accelerates the convergence. Fig. 11 shows the convergence and stability of the loss in (1). The variation of the loss value is reduced if EDIP is used. As a practical post-hoc strategy to overcome the instability of the DIP optimization scheme, the reconstructed image is taken as the network output at minimum loss.

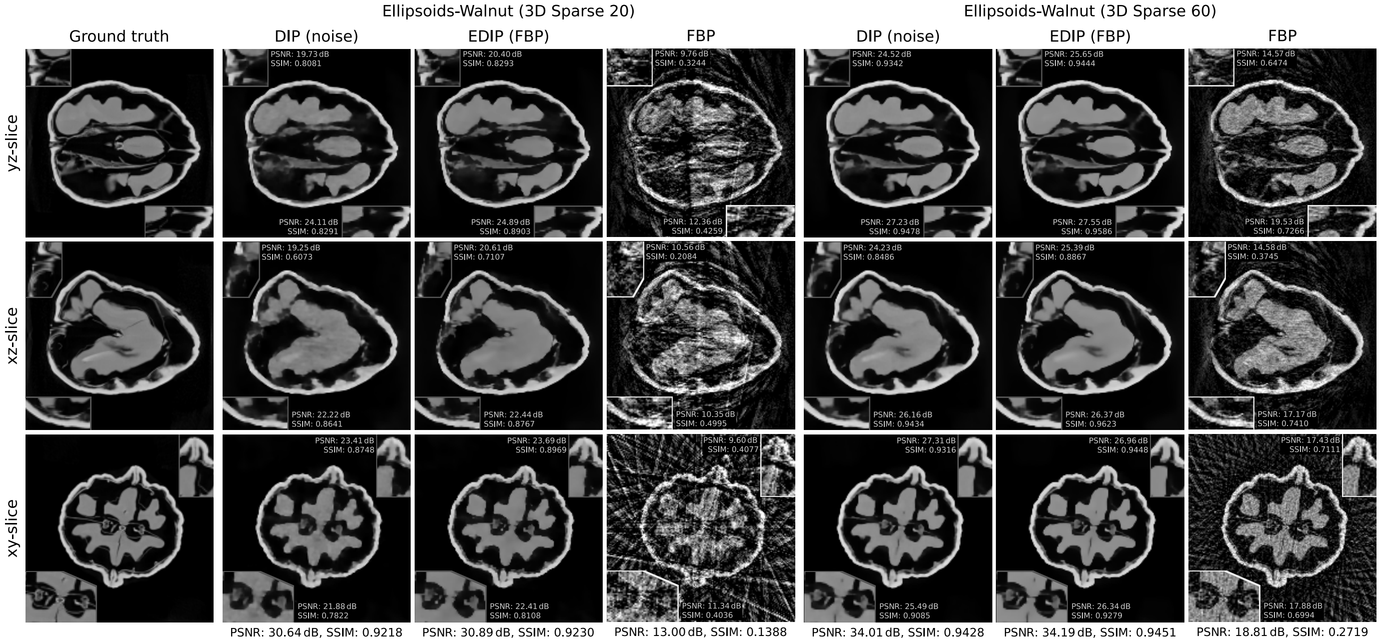

Ellipsoids-Walnut (3D Sparse 20)
Ellipsoids-Walnut (3D Sparse 60)
Ground truth
DIP (noise)
EDIP (FBP)
FBP
DIP (noise)
EDIP (FBP)
FBP
yz-slice
xz-slice
xy-slice
PSNR: 19.73 dB SSIM: 0.8081
PSNR: 24.11 dB SSIM: 0.8291
PSNR: 20.40 dB SSIM: 0.8293
PSNR: 24.89 dB SSIM: 0.8903
PSNR: 9.76 dB SSIM: 0.3244
PSNR: 12.36 dB SSIM: 0.4259
PSNR: 24.52 dB SSIM: 0.9342
PSNR: 27.23 dB SSIM: 0.9478
PSNR: 25.65 dB SSIM: 0.9444
PSNR: 27.55 dB SSIM: 0.9586
PSNR: 14.57 dB SSIM: 0.6474
PSNR: 19.53 dB SSIM: 0.7266
PSNR: 19.25 dB SSIM: 0.6073
PSNR: 22.22 dB SSIM: 0.8641
PSNR: 20.61 dB SSIM: 0.7107
PSNR: 22.44 dB SSIM: 0.8767
PSNR: 10.56 dB SSIM: 0.2084
PSNR: 10.35 dB SSIM: 0.4995
PSNR: 24.23 dB SSIM: 0.8486
PSNR: 26.16 dB SSIM: 0.9434
PSNR: 25.39 dB SSIM: 0.8867
PSNR: 26.37 dB SSIM: 0.9623
PSNR: 14.58 dB SSIM: 0.3745
PSNR: 17.17 dB SSIM: 0.7410
PSNR: 23.41 dB SSIM: 0.8748
PSNR: 21.88 dB SSIM: 0.7822
PSNR: 23.69 dB SSIM: 0.8969
PSNR: 22.41 dB SSIM: 0.8108
PSNR: 9.60 dB SSIM: 0.4077
PSNR: 11.34 dB SSIM: 0.4036
PSNR: 27.31 dB SSIM: 0.9316
PSNR: 25.49 dB SSIM: 0.9085
PSNR: 26.96 dB SSIM: 0.9448
PSNR: 26.34 dB SSIM: 0.9279
PSNR: 17.43 dB SSIM: 0.7111
PSNR: 17.88 dB SSIM: 0.6994
PSNR: 30.64 dB, SSIM: 0.9218
PSNR: 30.89 dB, SSIM: 0.9230
PSNR: 13.00 dB, SSIM: 0.1388
PSNR: 34.01 dB, SSIM: 0.9428
PSNR: 34.19 dB, SSIM: 0.9451
PSNR: 18.81 dB, SSIM: 0.2719

Fig. 9. 3D Walnut reconstruction of EDIP pretrained on ellipsoids dataset, compared to standard DIP, at three different slices.

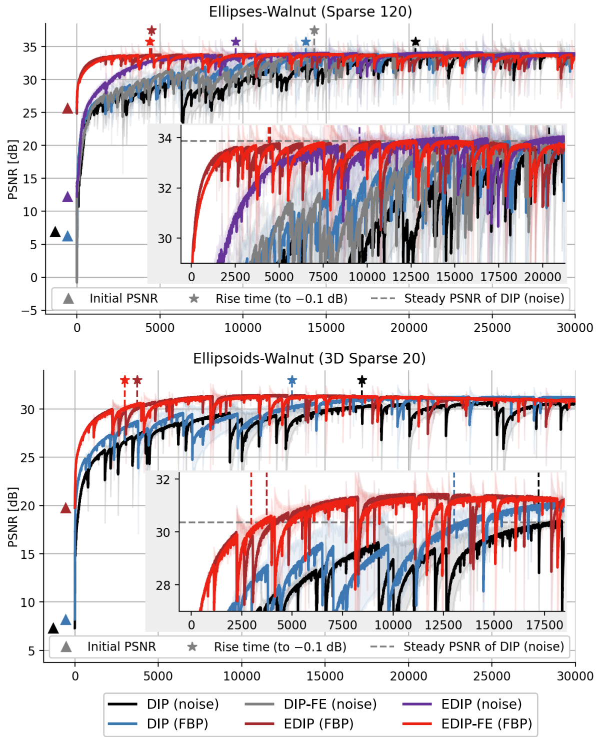


Fig. 10. The optimization of EDIP versus DIP for the Walnut reconstruction in 2D (top) and 3D (bottom). The symbols ⋆ and ▲ denote initial PSNR and rise time, respectively, and the horizontal dashed line indicates the steady PSNR of DIP (noise).

VII. Investigation of the Role of Pretraining

In this section, we first motivate why we use a standard pretraining strategy instead of resorting to more sophisticated schemes, and then we shed insight into the mechanism of knowledge transfer via pretraining, highlighting favorable as well as detrimental properties.

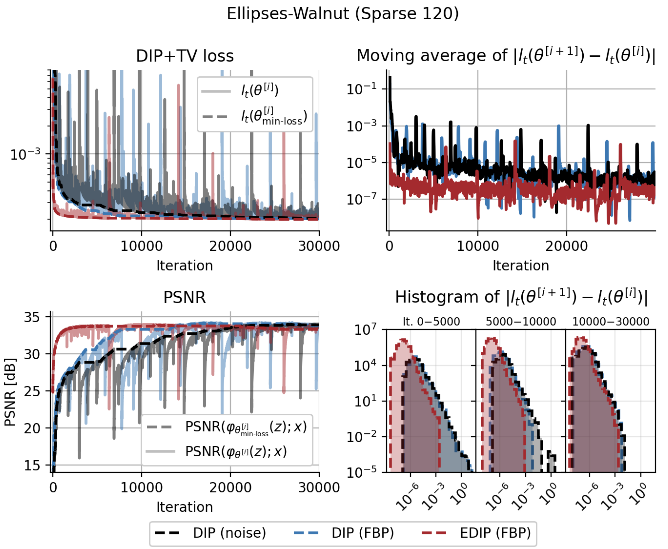


Fig. 11. The min-loss and PSNR computed with the min-loss output for Walnut Sparse 120 (left). Loss variation (i.e. $|l_t(\theta^{[i+1]}) - l_t(\theta^{[i]})|$) and respective histograms computed over three intervals (right). The moving average uses a window size of 100 iterations.

a) Standard, Adversarial, Meta?: In this work, we adopt the standard pretraining paradigm within our pretraining stage, as described in Sec. IV. The choice is informed by comparing standard pretraining, adversarial pretraining [51], [52] and model agnostic meta-learning (MAML) [53], [54] on the Lotus Sparse 20. Adversarial pretraining uses a projected gradient descent attack (PGD-L_2 [55], [56]). MAML-based pretraining obtains a parameters' configuration training on six different tasks, comprising three different image classes: ellipses, rectangles [57], and natural images from the PASCAL VOC segmentation dataset [58], as well as two different noise distributions: Gaussian and Poisson. We do not vary the forward operator A, since the pretraining stage is tailored to a known acquisition geometry; varying the structure of A (e.g., via sparsification) would only withhold from the model operator-specific knowledge, and introduce artifacts that are not expected to be found in the subsequent reconstruction tasks. We investigate whether the parameters' configurations, found with adversarial pretraining and MAML, lead to general representations adapting faster to the subsequent reconstruction problem. It is observed from Tab. IV that all three pretraining strategies lead to parameters' configurations that adapt to the subsequent reconstruction task with approximately similar speed-up. Even if the adaptation to the subsequent task shows on par properties, adversarial pretraining and MAML introduce a significant computational overhead, which we find unnecessary. The latter can be attributed to the facts that adversarial pretraining requires the inclusion of an inner loop optimization to design the attack (adding 62h to the wall-clock time); MAML's outer loop updates θ_{s}, while the inner one (with one step of stochastic gradient descent) adapts θ_{s} to a given task. MAML, instead, increases ($\times 5$) the overall VRAM required.

TABLE IV
QUANTITATIVE EVALUATION OF ALTERNATIVE PRETRAINING STRATEGIES FOR THE LOTUS ALONG WITH THE WALL-CLOCK TIME RECORDED ON A NVIDIA RTX 2080TI.

Ellipses-Lotus `Sparse 20`

	Rise time	(Max PSNR; iters)	(VRAM; batch size)	Time
EDIP (FBP)	195	(31.65; 981)	(5941 MiB; 32)	23h
Adv.-L_2-EDIP (FBP)	143	(31.24; 1175)	(6093 MiB; 32)	85h
MAML-EDIP (FBP)	545	(31.54; 1512)	(7949 MiB; 8)	31h

b) In Need to Amend: Figs. 7 and 8 show that the reconstructions obtained by directly deploying the pretrained network (i.e. $\varphi_{\theta_{\mathrm{s}}^*}$) on the FBP of the real-measured μCT data do enjoy good reconstructive properties, but the images tend to be overly-smooth and severely affected by ellipses-like artifacts, which are naturally present in the synthetic training dataset. Indeed, initializing the network's parameters to the pretrained configuration, on both Lotus and Walnut, shows a gain of 5.8 dB, and of 9.4 dB (`Sparse 120`), 6.7 dB (`3D Sparse 20`), 2 dB (`3D Sparse 60`) over the FBP. The pretrained model enjoys high input-robustness, and feature reuse plays a very important role in the EDIP reconstruction. However, the feature reuse mechanism leads to undesirable hallucinatory behaviors, as evidenced by the ellipses-like artifacts, which is a form of inductive biases induced by the synthetic image class. This also indicates the importance of properly designing the synthetic dataset used in the pretraining stage, from which the features are learned, and the strong dissimilarity between the synthetic training data and real test data may actually deteriorate the performance. In the supplementary materials, we showcase one potential pitfall of the "supervised pretraining + unsupervised fine-tuning" paradigm for DIP, resorting to synthetic data generated by a by far too specific and less diverse image class, i.e., human brain images for the supervised learning stage.

The knowledge enforced via the synthetic dataset needs to be properly amended so that the reconstructed images recover a more realistic texture. This is achieved at the fine-tuning stage by enforcing the data consistency. Amending the knowledge acquired via pretraining protects from hallucinations due to (inevitable) distributional shifts, thereby overcoming a well-known drawback of supervised learned reconstructors [59].

c) Investigating Feature Reuse: In a similar spirit to [60], we feed a noise image to EDIP (trained on pairs of FBP and ground truth image), which makes any visual features learned in the pretraining stage useless. This allows us to disentangle influencing factors involved in the fine-tuning stage. We consistently observe faster convergence of EDIP with respect to the standard DIP for the Lotus dataset. EDIP (fed with FBP) still results in faster convergence, which agree well with the intuition that decreasing feature reuse leads to diminishing benefits. Fig. 12 (left) shows that EDIP remolds the noise image differently compared to the standard DIP. The learned inductive biases prioritize reshaping the noise image as ellipse-like structures. The model makes an educated reconstruction. The features learned during pretraining are invariant of the input. The pretrained model is then adapted by enforcing data-consistency via (1).

On the Walnut, cf. Fig. 12 (right), the benefit of pretraining is less pronounced, if a noise image input is used. This might be due to the fact that the Walnut has a higher resolution and many more fine details, which are not present in the training dataset. Nonetheless, pretraining can still remold noise input into a walnut faster than DIP, yet the FBP input (used in the pretraining) is even more effective. These observations fully agree with that for Lotus.

d) Getting θ_{s}^* Right: The starting point θ_{s}^* of fine-tuning can impact the adaptation speed. A selection procedure of θ_{s}^* is desired to maximize transferable performance (e.g. speed-up). On the 2D setting, pretraining for more epochs (100 vs. 20) leads to a faster adaptation. This is clearly observed on the Lotus, possibly due to the in-distribution nature of the image class with respect to the ellipses dataset. However, on more complex tasks (`3D Sparse 20` and `3D Sparse 60`), extensive pretraining leads to overfitting the image class, and enforcing dataset-specific knowledge appears detrimental to the transfer. Fig. 13 shows that extensively pretraining U-Net for 2 epochs (i.e. 64k gradient updates with 32k ellipsoid volumes), albeit yielding the highest initial PSNR, leads to a sub-optimal convergence: the network output is effectively constrained, as an over-trained ϕ_{θ^*} after 2 epochs has little freedom to amend. This is also observed on the `3D Sparse 60` setting.

e) Spectral Evaluation: We propose a spectral analysis to understand the "education" by linearizing the non-linear forward map $F(\theta) = A\varphi_\theta(A^\dagger y_\delta)$ at θ_0:

$$F(\theta) = F(\theta_0) + F'(\theta_0)(\theta - \theta_0),$$

with $F'(\theta_0) = A\varphi'_{\theta_0} \in \mathbb{R}^{m\times p}$ with $\varphi'_{\theta_0} = \partial\varphi_\theta/\partial\theta|_{\theta=\theta_0} \in \mathbb{R}^{n\times p}$ denoting the Jacobian of the network's output w.r.t. θ. We use the subspace spanned by leading right singular vectors v_i of $F'(\theta_0)$ (i.e. with the largest singular values) as a faithful representation of the network's parameter space, which determines the dynamics of the learning process. Due to the high-dimensionality of the output and parameter spaces,

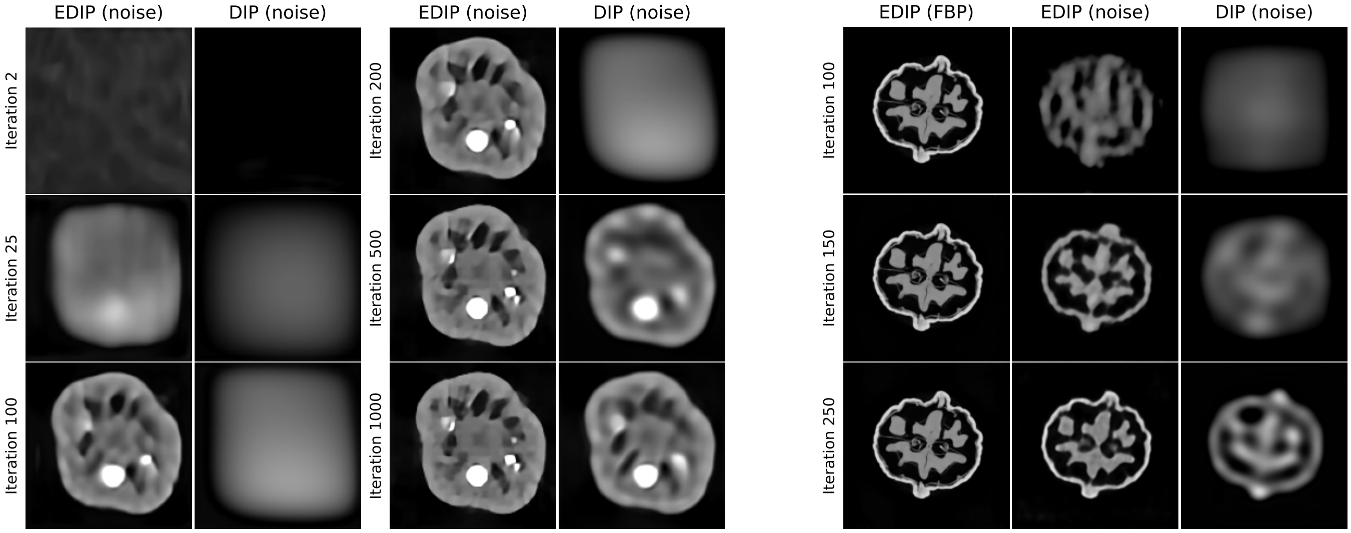


Fig. 12. Iterates collected throughout the EDIP/DIP reconstruction from Lotus `Sparse 20` (left) and Walnut `Sparse 120` (right), after different numbers of iterations. A video showing the reconstruction process is available at https://educateddip.github.io/docs.educated_deep_image_prior/.

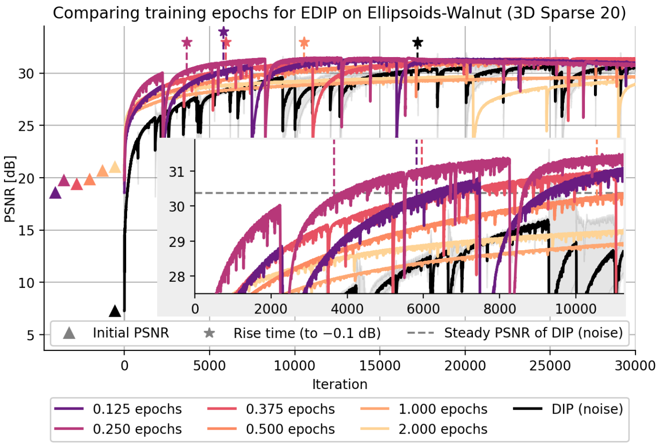


Fig. 13. The optimization of EDIP using parameters from different checkpoints for EDIP (FBP) on Walnut `3D Sparse 20`. The symbols $\star$ and $\blacktriangle$ denote initial PSNR and rise time, respectively, and the horizontal dashed line indicates the steady PSNR of DIP (noise).

Algorithm 1 rSVD for Linearized Forward Map

```
Require: the Jacobian matrix $F'(\theta_0)$, the target rank $\kappa$, and oversampling parameter $o$
Draw a $p\times(\ell=\kappa+o)$ Gaussian random matrix $\Omega=(\omega_{ij})$
Form $\bar{F}=F'(\theta_0)\Omega$
Construct an orthonormal basis $Q$ of range$(\bar{F})$ using QR decomposition
Form the matrix $B=Q^\top F'(\theta_0)$
Compute the SVD of $B=W\tilde{\Sigma}_\ell\tilde{V}_\ell$
Return $\tilde{\Sigma}_\kappa, \tilde{V}_\kappa$
```

directly computing φ'_{θ_0} is intractable. We approximate the first ℓ singular vectors of $F'(\theta_0)$ via randomized singular value decomposition (rSVD) [61], [62], and proceed in two steps (cf. Algorithm 1):

Stage #1: Randomized Range Finder. To construct a subspace capturing most of the action of $F'(\theta_0)$, we draw a Gaussian random matrix $\Omega\in\mathbb{R}^{p\times\ell}$ and form $\bar{F}=F'(\theta_0)\Omega\in\mathbb{R}^{m\times\ell}$. To avoid the direct evaluation of φ'_{θ_0}, for any column ω of Ω, we use a finite difference approximation: $\varphi'_{\theta_0}\omega=(\varphi_{\theta_0+\epsilon\omega}-\varphi_{\theta_0-\epsilon\omega})/(2\epsilon)$, where $\epsilon>0$ is a small constant. Then we find an orthonormal matrix $Q\in\mathbb{R}^{m\times\ell}$ for the range of $\bar{F}$, using the standard QR factorization [61], [62].

Stage #2: Direct SVD. Next we construct a low-rank matrix $B=Q^\top F'(\theta_0)\in\mathbb{R}^{\ell\times p}$, or equivalently, $B^\top=F'(\theta_0)^\top Q$, which can be computed via backpropagation, and then approximate the singular values and the right singular vectors of $F'(\theta_0)$ by that of $B\approx U\Sigma V^\top$ (with the last few discarded as oversampling: default choice 5). Since the size of $B\in\mathbb{R}^{\ell\times p}$ is much smaller than that of $F'(\theta_0)$, a direct SVD computation is indeed feasible.

In the analysis, we use the 995 leading singular values and the corresponding right singular vectors, which are used to represent the parameters. We investigate EDIP and DIP, both receiving the FBP as the input, and respectively approximate the singular vectors of the Jacobian, evaluated at three checkpoints during the fine-tuning stage ($\theta^{\rm init}$, $\theta^{[100]}$, $\theta^{\rm conv}$). Fig. 14 summarizes our empirical findings, showing the right singular values component-wise plots and Hoyer measure of sparsity [63], [64]. Hoyer measure takes a value 0 if the vector is dense (i.e. all components are equal and non-zero) and 1 if it is 1-sparse. The histogram is computed for the two sets of singular vectors, i.e., $\{v_1,\ldots,v_{20}\}$ and $\{v_{976},\ldots,v_{995}\}$, separately, in order to examine the behavior at the different frequency bands. For DIP, the singular vectors are equally distributed throughout the parameter space (at $\theta^{\rm init}$) and across different singular values. During the fine-tuning stage, we observe a "relevance shift" towards the decoder's parameters (at $\theta^{[100]}$ and at $\theta^{\rm conv}$, respectively), which is attributed to the fact that the heavy-lifting of representing the target image is actually done by the decoder. This is also consistent with our experimental findings: EDIP-FE shows very similar reconstruction properties to EDIP. For EDIP, pretraining enforces a hierarchical structure

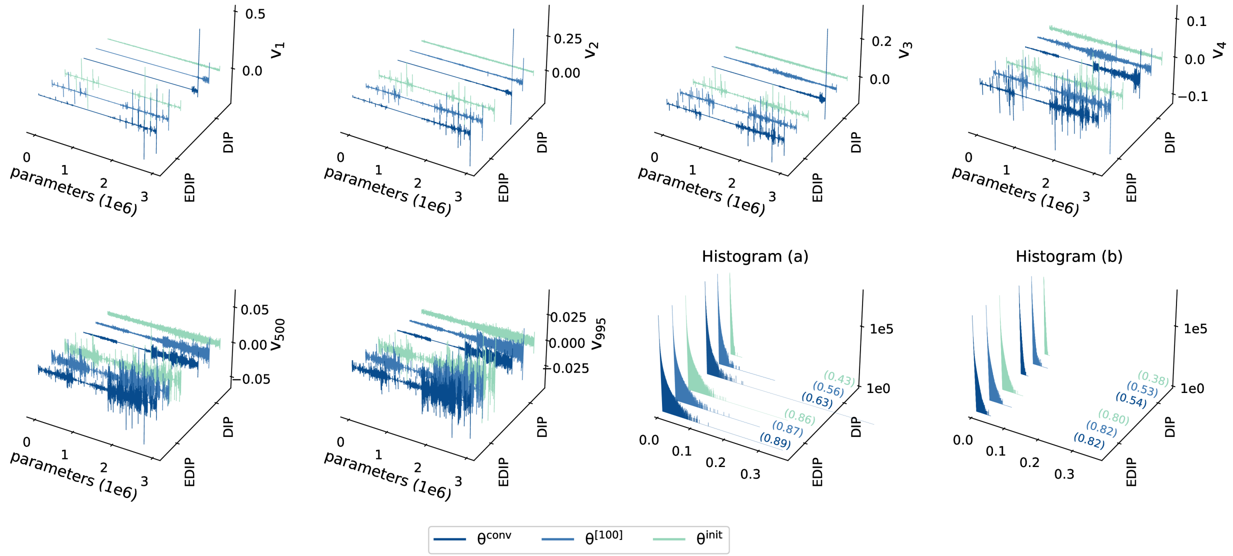


Fig. 14. The evolution of right singular vectors of the linearized forward map (i.e., the Jacobian) w.r.t. the network parameters θ for EDIP (FBP) versus DIP (FBP) on Lotus `Sparse 20` dataset. The parameters are ordered like they occur in the network, i.e. lower positions on the parameters axis refer to the encoder while higher positions refer to the decoder. (a) and (b) show mean histograms for the right singular vectors $v_1, \ldots, v_{20}$ and $v_{976}, \ldots, v_{995}$, which represent the low-frequency and high-frequency bands of the singular vectors, respectively; the numbers in brackets denote Hoyer measure of sparsity [63], [64].

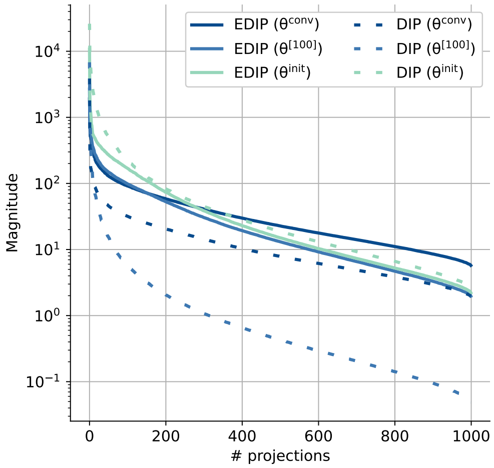


Fig. 15. The singular values of the linearized forward map (i.e., the Jacobian) w.r.t. the network parameters θ, at θ^{conv} and θ^{init} for EDIP (FBP) and DIP (noise) on Lotus `Sparse 20` data.

(i.e. a relevance shift towards the decoder's parameters), and again sparsity is clearly observed after pretraining. Pretraining strongly promotes sparsity in the basis of the parameter space, which is further promoted in the fine-tuning stage. This is observed in both low and high frequency bands. It is worth noting that even though individual singular vectors exhibit sparsity, the parameter vector θ does not necessary exhibit a very high level of sparsity, since the linear combination might spoil it. The emerging sparsity during pretraining may facilitate pruning the network, which however is still to be systematically explored.

Interestingly, pretraining also induces a shift in the singular values spectrum, and the overall behavior does not vary much during adaptation, cf. Fig. 15. In contrast, for DIP, the shift is quite dramatic in terms of the magnitude, as well as the number of singular values larger than a given threshold. This may offer an explanation to the very different dynamics of the optimization scheme for the pretrained model and the model trained from scratch: in the linearized regime, the singular value spectrum essentially determines the dynamics of gradient type algorithms (along with the learning rate), and the dramatic shift of the singular value spectrum of the DIP Jacobian may have contributed to the undesirable unsteady convergence behavior of DIP and indicates the necessity of carefully tuning the learning rate schedule in order to achieve a stable convergence behavior.

VIII. Conclusions

Our work advances unsupervised deep learning-based tomographic reconstruction. We develop a two-stage learning paradigm for accelerating DIP in image reconstruction. It consists of a supervised pretraining stage on a simulated dataset to educate DIP and then a fine-tuning stage which adapts the network parameters to a single test image. The extensive experimental evaluation clearly shows that pretraining on simulated data can significantly speed up, and stabilize DIP reconstruction for 2D / 3D real-measured sparse-view μCT. The empirical study also indicates that the pretraining stage can facilitate learning a suitable feature representation, and that adapting only the decoder's parameters during the fine-tuning stage is sufficient to ensure good reconstruction accuracy. The novel spectral analysis of the linearized model indicates a strong correlation of the sparsity pattern with the pretraining, and a drastically different shift of the singular values spectrum for the standard DIP and the educated version.

There are several avenues for further research. First, there are other techniques for learning a good initialization for neural networks, e.g., model-agnostic meta-learning (MAML)

[53] and adversarial pretraining [51], [52]. These strategies are also promising, but their full potentials are yet to be explored within the context of DIP reconstruction. In the spirit of ANIL (Almost No Inner Loop) [54], we would suggest using a variant that simplifies the inner loop optimization so as to improve the scalability of MAML. Second, given the emerging sparsity pattern in singular vectors, it is natural to ask whether one can exploit for even faster adaptation, e.g., via pruning or optimizing in low-dimensional subspaces. Third, the proposal utilizes the specific forward operator in the pretraining stage, and hence the pretrained neural network is specialized, where specialization to the target task is believed to be helpful. However, addressing multiple settings (e.g., different imaging modalities and multiple image classes) simultaneously is of course of interest.

Acknowledgment

R.B. was supported by the i4health PhD studentship (UK EPSRC EP/S021930/1), and by The Alan Turing Institute under the UK EPSRC grant EP/N510129/1. J.L., M.S., and A.D. were funded by the German Research Foundation (DFG; GRK 2224/1). J.L. and M.S. additionally acknowledge support from the DELETO project funded by the Federal Ministry of Education and Research (BMBF, project number 05M20LBB). A.D. further acknowledges support from the Klaus Tschira Stiftung via the project MALDISTAR (project number 00.010.2019). A.H. acknowledges funding from the Academy of Finland projects 338408, 336796, 334817. P.M. was supported by the Sino-German Center for Research Promotion (CDZ) via the Mobility Programme 2021: Inverse Problems – Theories, Methods and Implementations (IP–TMI). The research of B.J. is supported by UK EPSRC grants EP/T000864/1 and EP/V026259/1.

References

[1] H. W. Engl, M. Hanke, and A. Neubauer, *Regularization of Inverse Problems*. Kluwer, Dordrecht, 1996. 1, 2

[2] O. Scherzer, M. Grasmair, H. Grossauer, M. Haltmeier, and F. Lenzen, *Variational Methods in Imaging*. New York, NY: Springer, 2009. 1, 2

[3] K. Ito and B. Jin, *Inverse Problems: Tikhonov Theory and Algorithms*. World Scientific, Hackensack, NJ, 2015. 1, 2

[4] G. Ongie, A. Jalal, R. G. Baraniuk, C. A. Metzler, A. G. Dimakis, and R. Willett, "Deep learning techniques for inverse problems in imaging," *IEEE J. Sel. Areas Inform. Theory*, vol. 1, no. 1, pp. 39–56, 2020. 1

[5] S. Arridge, P. Maass, O. Öktem, and C.-B. Schönlieb, "Solving inverse problems using data-driven models," *Acta Numer.*, vol. 28, pp. 1–174, 2019. 1

[6] D. O. Baguer, J. Leuschner, and M. Schmidt, "Computed tomography reconstruction using deep image prior and learned reconstruction methods," *Inverse Problems*, vol. 36, no. 9, p. 094004, 2020. 1, 2, 3, 4, 13

[7] D. Gilton, G. Ongie, and R. Willett, "Model adaptation for inverse problems in imaging," *IEEE Trans. Comput. Imag.*, vol. 7, pp. 661–674, 2021. 1, 2

[8] R. Barbano, Z. Kereta, A. Hauptmann, S. R. Arridge, and B. Jin, "Unsupervised knowledge-transfer for learned image reconstruction," *Inverse Problems*, vol. 38, no. 10, p. 104004, 2022. 1

[9] S. Lunz, A. Hauptmann, T. Tarvainen, C.-B. Schönlieb, and S. Arridge, "On learned operator correction in inverse problems," *SIAM J. Imag. Sci.*, vol. 14, no. 1, pp. 92–127, 2021. 1

[10] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep image prior," in *CVPR*, 2018, pp. 9446–9454. 1, 2, 3

[11] T. Knopp and M. Grosser, "Warmstart approach for accelerating deep image prior reconstruction in dynamic tomography," Proceedings of Machine Learning Research, Medical Imaging with Deep Learning 2022, 13 pp., 2022. 1, 2

[12] M. Z. Darestani and R. Heckel, "Accelerated MRI with un-trained neural networks," pp. 724–733, 2021. 1

[13] K. Gong, C. Catana, J. Qi, and Q. Li, "PET image reconstruction using deep image prior," *IEEE Trans. Med. Imag.*, vol. 38, no. 7, pp. 1655–1665, 2019. 1

[14] J. Cui, K. Gong, N. Guo, C. Wu, K. Kim, H. Liu, and Q. Li, "Populational and individual information based PET image denoising using conditional unsupervised learning," *Phys. Med. & Biol.*, vol. 66, no. 15, p. 155001, 2021. 1

[15] J. Cui, K. Gong, N. Guo, C. Wu, X. Meng, K. Kim, K. Zheng, Z. Wu, L. Fu, B. Xu *et al.*, "PET image denoising using unsupervised deep learning," *Eur. J. Nuclear Med. Mol. Imag.*, vol. 46, no. 13, pp. 2780–2789, 2019. 1

[16] S. Barutcu, D. Gürsoy, and A. K. Katsaggelos, "Compressive ptychography using deep image and generative priors," Preprint, arXiv:2205.02397, 2022. 1

[17] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," *IEEE Signal Proc. Mag.*, vol. 38, no. 2, pp. 18–44, 2021. 1

[18] A. Hauptmann, F. Lucka, M. Betcke, N. Huynh, J. Adler, B. Cox, P. Beard, S. Ourselin, and S. Arridge, "Model-based learning for accelerated, limited-view 3-d photoacoustic tomography," *IEEE Trans. Med. Imag.*, vol. 37, no. 6, pp. 1382–1393, 2018. 1

[19] J. Leuschner, M. Schmidt, P. S. Ganguly, V. Andriiashen, S. B. Coban, A. Denker, D. Bauer, A. Hadjifaradji, K. J. Batenburg, P. Maass, and M. van Eijnatten, "Quantitative comparison of deep learning-based image reconstruction methods for low-dose and sparse-angle CT applications," *J. Imag.*, vol. 7, no. 3, p. 44, 2021. 1

[20] S. Azizi, B. Mustafa, F. Ryan, Z. Beaver, J. Freyberg, J. Deaton, A. Loh, A. Karthikesalingam, S. Kornblith, T. Chen *et al.*, "Big self-supervised models advance medical image classification," *Preprint, arXiv:2101.05224*, 2021. 1

[21] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," *Preprint, arXiv:1801.06146*, 2018. 1

[22] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," in *ICML*, 2014, pp. 647–655. 1

[23] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*. Springer, 2015, pp. 234–241. 2

[24] Y. Jo, S. Y. Chun, and J. Choi, "Rethinking deep image prior for denoising," ICCV 2021, arXiv:2108.12841, 2021. 2

[25] H. Wang, T. Li, Z. Zhuang, T. Chen, H. Liang, and J. Sun, "Early stopping for deep image prior," Preprint, arXiv:2112.06074, 2021. 2

[26] R. Heckel and P. Hand, "Deep decoder: Concise image representations from untrained non-convolutional networks," in *ICLR*, 2019. 2

[27] S. Dittmer, T. Kluth, P. Maass, and D. Otero Baguer, "Regularization by architecture: A deep prior approach for inverse problems," *J. Math. Imag. Vision*, vol. 62, no. 3, pp. 456–470, 2020. 2

[28] Z. Cheng, M. Gadelha, S. Maji, and D. Sheldon, "A Bayesian perspective on the deep image prior," in *CVPR*, 2019, pp. 5443–5451. 2

[29] J. Liu, Y. Sun, X. Xu, and U. S. Kamilov, "Image restoration using total variation regularized deep image prior," in *ICASSP*, 2019. 2, 3

[30] G. Mataev, P. Milanfar, and M. Elad, "DeepRED: Deep image prior powered by RED," in *ICCV Workshops*, Oct 2019. 2

[31] R. Heckel and M. Soltanolkotabi, "Compressive sensing with untrained neural networks: Gradient descent finds the smoothest approximation," in *Proceedings of the 37th International Conference on Machine Learning, PMLR 119*, 2020, pp. 4149–4158. 2

[32] C. Arndt, "Regularization theory of the analytic deep prior approach," *Inverse Problems*, vol. 38, no. 11, p. 115005, 2022. 2

[33] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *CVPR*, 2016, pp. 779–788. 2

[34] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," in *Advances in Neural Inf. Process. Syst.*, 2015, pp. 91–99. 2

[35] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015, pp. 3431–3440. 2

[36] K. He, R. Girshick, and P. Dollar, "Rethinking imagenet pre-training," in *ICCV*, 2019, pp. 4918–4927. 2

[37] M. Raghu, C. Zhang, J. Kleinberg, and S. Bengio, “Transfusion: Understanding transfer learning for medical imaging,” in *Proceedings of the 33rd Neural Information Processing Systems*, 2019, pp. 3347–3357. 2

[38] Y. Han, J. Yoo, H. H. Kim, H. J. Shin, K. Sung, and J. C. Ye, “Deep learning with domain adaptation for accelerated projection-reconstruction MR,” *Magn. Reson. Med.*, vol. 80, no. 3, pp. 1189–1205, 2018. 2

[39] S. U. H. Dar, M. Özbey, A. Çatlı, and B. T. Çukur, “A transfer-learning approach for accelerated mri using deep neural networks,” *Magn Reson Med*, vol. 84, no. 2, pp. 663–685, 2020. 2

[40] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *3rd ICLR*, Y. Bengio and Y. LeCun, Eds., 2015. 3

[41] X. Pan, E. Y. Sidky, and M. Vannier, “Why do commercial CT scanner still employ traditional, filtered back-projection for image reconstruction?” *Inverse Problems*, vol. 25, no. 12, pp. 123 009, 36, 2009. 3

[42] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser, “Deep convolutional neural network for inverse problems in imaging,” *IEEE Trans. Image Proc.*, vol. 26, no. 9, pp. 4509–4522, 2017. 3

[43] J. Adler, H. Kohr, A. Ringh, J. Moosmann, S. Banert, M. J. Ehrhardt, G. R. Lee, K. Niinimäki, B. Gris, O. Verdier, J. Karlsson, G. Zickert, W. J. Palenstijn, O. Öktem, C. Chen, H. A. Loarca, and M. Lohmann, “Operator Discretization Library (ODL),” 2018, *Zenodo*. 3

[44] T. A. Bubba, A. Hauptmann, S. Huotari, J. Rimpeläinen, and S. Siltanen, “Tomographic X-ray data of a lotus root filled with attenuating objects,” *Preprint, arXiv:1609.07299*, 2016. 4

[45] H. Der Sarkissian, F. Lucka, M. van Eijnatten, G. Colacicco, S. B. Coban, and K. J. Batenburg, “Cone-Beam X-Ray CT Data Collection Designed for Machine Learning: Samples 1-8,” 2019, *Zenodo*. [Online]. Available: https://doi.org/10.5281/zenodo.2686726 4

[46] L. A. Feldkamp, L. C. Davis, and J. W. Kress, “Practical cone-beam algorithm,” *J. Opt. Soc. Am. A*, vol. 1, no. 6, pp. 612–619, 1984. 4

[47] A. Hendriksen, D. Schut, W. J. Palenstijn, N. Viganò, J. Kim, D. Pelt, T. van Leeuwen, and K. J. Batenburg, “Tomosipo: Fast, flexible, and convenient 3D tomography for complex scanning geometries in Python,” *Optics Expr.*, Oct 2021. 4

[48] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, “3d eu-net: learning dense volumetric segmentation from sparse annotation,” in *MICCAI*. Springer, 2016, pp. 424–432. 4

[49] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: From error visibility to structural similarity,” *IEEE Trans. Image Proc.*, vol. 13, no. 4, pp. 600–612, 2004. 4

[50] L. A. Shepp and B. F. Logan, “The Fourier reconstruction of a head section,” *IEEE Trans. Nuclear Sci.*, vol. 21, no. 3, pp. 21–43, 1974. 5

[51] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” *International Conference on Learning Representations*, 2015. 7, 11

[52] M. Yi, L. Hou, J. Sun, L. Shang, X. Jiang, Q. Liu, and Z. Ma, “Improved ood generalization via adversarial training and pretraing,” in *International Conference on Machine Learning*. PMLR, 2021, pp. 11 987–11 997. 7, 11

[53] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in *International Conference on Machine Learning (ICML)*, 2017, pp. 1126–1135. 7, 11

[54] A. Raghu, M. Raghu, S. Bengio, and O. Vinyals, “Rapid learning or feature reuse? towards understanding the effectiveness of maml,” in *International Conference on Learning Representations*, 2019. 7, 11

[55] N. Carlini and D. Wagner, “Towards evaluating the robustness of neural networks,” in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 39–57. 7

[56] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” *International Conference on Learning Representations*, 2018. 7

[57] R. Barbano, J. Leuschner, J. Antorán, B. Jin, and J. M. Hernández-Lobato, “Bayesian experimental design for computed tomography with the linearised deep image prior,” *arXiv preprint arXiv:2207.05714*, 2022. 7

[58] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (voc) challenge,” *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, 2010. 7

[59] V. Antun, F. Renna, C. Poon, B. Adcock, and A. C. Hansen, “On instabilities of deep learning in image reconstruction and the potential costs of AI,” *Proc. Nat. Acad. Sci.*, vol. 117, no. 48, pp. 30 088–95, 2020. 8

[60] B. Neyshabur, H. Sedghi, and C. Zhang, “What is being transferred in transfer learning?” in *Advances in Neural Information Processing Systems*, vol. 33, 2020. 8

[61] N. Halko, P.-G. Martinsson, and J. A. Tropp, “Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions,” *SIAM Rev.*, vol. 53, no. 2, pp. 217–288, 2011. 9

[62] J. A. Tropp, “Randomized algorithms for matrix computations,” Caltech CMS Lecture Notes 2019-01, Pasadena, July 2019. 9

[63] P. O. Hoyer, “Non-negative matrix factorization with sparseness constraints,” *J. Mach. Learn. Res.*, vol. 5, pp. 1457–1469, 2004. 9, 10

[64] N. Hurley and S. Rickard, “Comparing measures of sparsity,” *IEEE Trans. Inform. Theory*, vol. 55, no. 10, pp. 4723–4741, 2009. 9, 10

Supplementary Material A
μCT Measurement data

A. Cone-Beam Geometry

On the Lotus root, we employ the sparse matrix provided with the dataset. For the 2D Walnut setting, a sparse matrix resembling the 2D cone-beam projection is constructed from the ASTRA geometry, by selecting a single volume slice, and a suitable subset of the 3D cone-beam projection lines. This is a non-standard 2D fan-beam setting: (i) the rotation axis is slightly tilted; (ii) the voxels / pixels are weighted according to the 3D projections, which differs from the 2D projection weighting. Specifically, in the integration of the beams for each detector "pixel", the contributing area / interval is spreading in two vs. one dimension(s) with increasing distance from the source, so the beam density decreases antiproportionally to the squared distance vs. antiproportionally to the distance. For the 3D Walnut settings, ASTRA's direct projection routines are employed via tomosipo. The backward gradients are approximated by back-projection. The geometry definition has been adapted to match the sub-sampling applied to the volume and the measurements.

B. X-ray Walnut Details

From the collection of 42 Walnuts, we consider measurements of Walnut 1 taken with source position (or orbit) 2. The slice with offset $+3\,\mathrm{px}$ from the middle slice (i.e. zero-based index 253) is selected for the 2D reconstruction task. A subset of projection values is determined from the provided ASTRA geometry by computing the 3D forward projection of a mask, containing ones for the selected 2D slice and zeros for all other voxels. We choose one single detector row per column and angle with maximum intensity. A sparse matrix representing the forward projection is constructed from the ASTRA forward projection routine for each unit vector, for which the transposed matrix gives an exact adjoint of the Jacobian, used in computing the gradient of (1). The more efficient ASTRA back-projection routine is not directly applicable due to the pseudo-2D geometry: some of the excluded detector rows close to the selected ones contribute to the selected 2D slice in the back-projection. Another workaround (without matrix assembly) is to copy the measurement values from the selected rows to the neighboring rows (a.k.a. edge-mode padding); we use this to compute approximate FDK reconstructions. For computing the gradient of the data fitting term in (1), using the padding followed by the back-projection via ASTRA leads to degraded results, so we use the sparse matrix multiplication instead, which yields accurate gradients.

The implementation and the sparse matrix are available at https://educateddip.github.io/docs.educated_deep_image_prior/.

Supplementary Material B
Methodology

A. 2D Network architecture

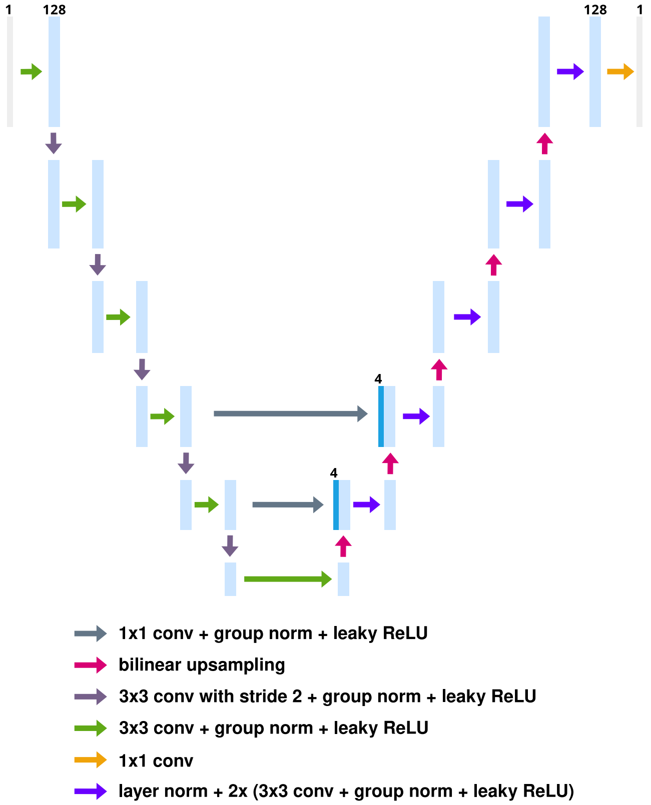

1
128
128
1
4
4
1x1 conv + group norm + leaky ReLU
bilinear upsampling
3x3 conv with stride 2 + group norm + leaky ReLU
3x3 conv + group norm + leaky ReLU
1x1 conv
layer norm + 2x (3x3 conv + group norm + leaky ReLU)

Fig. 16. The architecture of the U-Net used for the 2D experiments. Each light-blue bar corresponds to a multi-channel feature map. Arrows denote the different operations. The number of channels is set to 128 at every scale.

Figure 16 shows the network architecture used. We adopted the architecture proposed by [6], with the only difference being that we replace batch-normalization layers with group-normalization layers.

See also Figure 4 in the main text showing the U-Net architecture used for the 3D experiments.

B. The Loss

Our DIP implementation uses the loss function

$$l'_{\mathrm{t}}(\theta) := \frac{1}{m}\|A\varphi_\theta(z) - y_\delta\|_2^2 + \gamma'\,\mathrm{TV}(\varphi_\theta(z)),$$

with the anisotropic total variation penalty $\mathrm{TV}(x) = \|\nabla_h x\|_1 + \|\nabla_v x\|_1$, where m is the number of detector pixels (length of y_δ) and ∇_h and ∇_v are the discrete difference operators in the horizontal and vertical directions, respectively.

C. Hyperparameter Search

For each setting, suitable hyperparameters for DIP (noise) are selected by grid search. While the learning rate $1\mathrm{e}{-4}$ is (near) optimal in all cases, the TV-regularization parameter γ' varies both with the μCT geometry and between validation data (i.e. Shepp-Logan phantom, simulated data) and test data (i.e. Lotus or Walnut, real data).

TABLE V
HYPERPARAMETERS FOR (E)DIP ON VALIDATION AND TEST DATA.

Validation	Learn. rate	γ'	Iters.
Lotus `Sparse 20`	1e−4	4e−5	37 500
Lotus `Limited 45`	1e−4	1e−6	15 000
↪ EDIP (FBP)	1e−4	4e−6	10 000
Walnut `Sparse 120`	1e−4	2e−7	50 000
Test	**Learn. rate**	**γ'**	**Iters.**
Lotus `Sparse 20`	1e−4	1e−4	10 000
Lotus `Limited 45`	1e−4	6.5e−5	10 000
Walnut `Sparse 120`	1e−4	2e−7	30 000
↪ EDIP[-FE] (noise) pretrained on ellipses	5e−4 to 1e−4	2e−7	30 000
Walnut `3D Sparse 20`	1e−4	1e−1	30 000
Walnut `3D Sparse 60`	5e−5	1e−1	60 000

The hyperparameters used for DIP and EDIP are listed in Table V. The parameters are fine-tuned on DIP (noise), except for the override values specified in the rows starting with "↪". For only two cases, we observe the hyperparameters that are optimal for DIP (noise) to be severely sub-optimal for EDIP. For instance, no speed-up is observed for EDIP (noise), applied to the Walnut `Sparse 120`, after pretraining on the ellipses dataset, if the default learning rate 1e−4 is used; while a higher learning rate leads to an unstable optimization. A "warm-up" learning rate scheduling with an initial learning rate of 5e−4, which is linearly decreased to 1e−4 over the first 5k iterations reveals a substantial speed-up. We use the same learning rate scheduling with DIP (noise), but fail to observe any improvement. Similarly, we observe that validating on the Shepp-Logan phantom for the Lotus `Limited 45` setting requires the regularization parameter γ' to be increased to 4e−6 (instead of 1e−6) for EDIP (FBP) to converge.

TABLE VI
HYPERPARAMETERS FOR LOTUS GOLD-STANDARD REFERENCE RECONSTRUCTION.

Reference	Learn. rate	γ'	Iters.
Lotus (full 120) TV	1e-3	5e−5	1000

TABLE VII
HYPERPARAMETERS FOR TV BASELINES ON TEST DATA.

Test	Learn. rate	γ'	Iters.
Lotus `Sparse 20` TV	5e−4	1e−4	5000
Lotus `Limited 45` TV	5e−4	4e−5	5000
Walnut `Sparse 120` TV	5e−4	4e−7	10 000
Walnut `3D Sparse 20` TV	5e−4	2e−1	5000
Walnut `3D Sparse 60` TV	5e−4	1e−1	5000

SUPPLEMENTARY MATERIAL C
EXTENDED EXPERIMENTAL RESULTS

Here we report additional details about the experiments.

A. *The Lotus (Continued)*

We also include a limited-view setting, named Lotus `Limited 45`: 45 angles, range $[0, 135°)$ in steps of 3°. Fig. 17 shows exemplary reconstructions on the test-fold of the synthetic datasets used for pretraining, for both `Sparse 20` and `Limited 45`. The FBP suffers severe streak artifacts, but the trained U-Net can recover the shapes well.

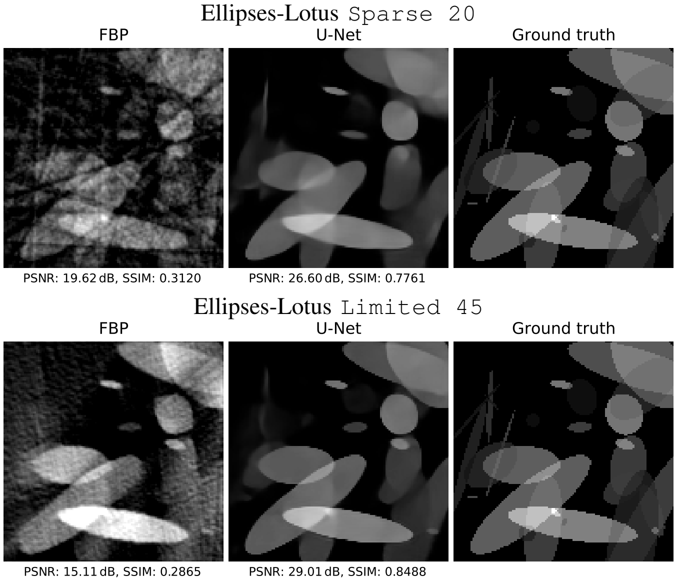

Ellipses-Lotus Sparse 20
FBP
U-Net
Ground truth
PSNR: 19.62 dB, SSIM: 0.3120
PSNR: 26.60 dB, SSIM: 0.7761
Ellipses-Lotus Limited 45
FBP
U-Net
Ground truth
PSNR: 15.11 dB, SSIM: 0.2865
PSNR: 29.01 dB, SSIM: 0.8488

Fig. 17. Exemplary reconstructions from the synthetic training datasets for Lotus `Sparse 20` and `Limited 45`.

The PSNR convergence of EDIP on Lotus root for the `Limited 45` setting is shown in Fig. 18; the reconstructions are reported in Fig. 19. These numerical results indicate analogous conclusions as for the case of `Sparse 20`.

Table VIII reports overall tabular results for Lotus `Sparse 20` and Lotus `Limited 45`. Rise time is defined to be the minimal number of iterations after which the PSNR reaches steady PSNR of DIP (noise) minus 0.1 dB. Both maximum PSNR and steady PSNR are computed using the iteration-wise median PSNR history over the 5 repeated runs (varying the random seed). For steady PSNR, the median value of the median PSNR history over the last 5k iterations is considered. The convergence of TV is observed to be very stable, and we report the final PSNR. Initial PSNR is the mean value over the 5 repeated runs.

It is observed that pretraining can substantially accelerate and stabilize the convergence of DIP. The acceleration factor is more substantial, when considering the FBP as input. The maximum PSNR (Max. PSNR) and steady PSNR suggest that pretraining also improves the reconstruction quality. The performance of EDIP-FE is largely comparable to EDIP.

Fig. 20 shows the convergence of the loss in (1) and of the PSNR, where the PSNR is computed using the network output with minimum loss reached until the current iteration. Using the minimum loss output is a practical way to overcome the instability of DIP optimization, clearly observed in the plots with the raw data in the main analysis. Pretraining greatly accelerates and stabilizes subsequent unsupervised training of EDIP, when compared to the standard DIP. This indicates a more favorable optimization landscape of EDIP / EDIP-FE than that of DIP. A stable convergence in practice is important for designing stopping rules for DIP / EDIP.

TABLE VIII
QUANTITATIVE EVALUATION FOR LOTUS `Sparse 20` AND LOTUS `Limited 45` WITH EDIP BEING PRETRAINED ON ELLIPSES DATA.

Ellipses-Lotus	`Sparse 20`				`Limited 45`			
	Rise time	(Max PSNR; iters)	Steady PSNR	Init PSNR	Rise time	(Max PSNR; iters)	Steady PSNR	Init PSNR
DIP (noise)	3848	(31.17; 8846)	31.10	11.17	5470	(29.85; 9690)	29.69	11.17
DIP (FBP)	3622	(31.25; 8813)	31.17	11.33	5419	(29.84; 8898)	29.69	11.32
DIP-FE (noise)	6118	(31.10; 9818)	31.00	11.17	5142	(29.82; 8884)	29.69	11.17
DIP-FE (FBP)	4516	(31.19; 7677)	31.13	11.33	5056	(29.83; 9891)	29.67	11.32
EDIP (FBP)	195	(31.65; 981)	31.21	27.04	524	(29.83; 2734)	29.68	27.55
EDIP (noise)	723	(31.53; 3548)	31.39	14.28	682	(29.94; 4445)	29.80	14.34
EDIP-FE (FBP)	226	(31.59; 1421)	31.26	27.04	245	(29.85; 5533)	29.72	27.55
EDIP-FE (noise)	1414	(31.46; 4278)	31.39	14.28	1279	(29.95; 7095)	29.86	14.34
TV	–	–	30.73	–	–	–	29.62	–

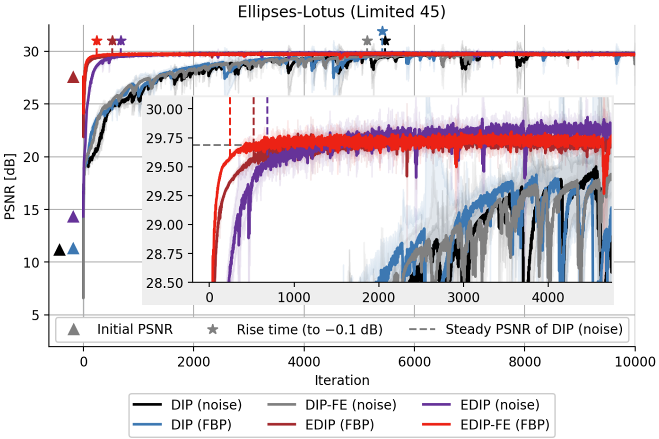
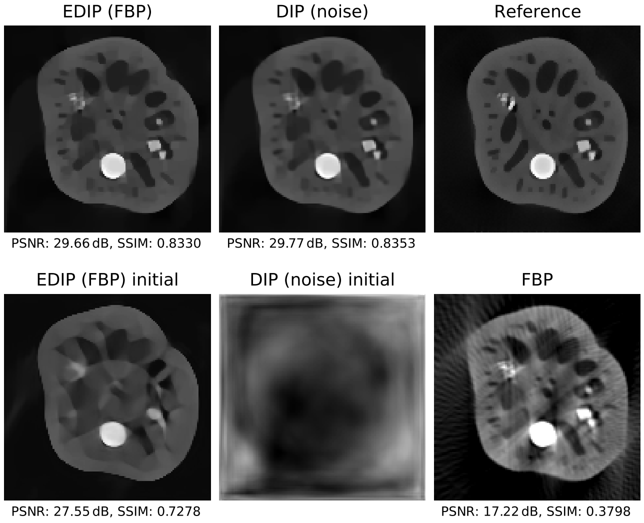


Fig. 18. The optimization of EDIP versus DIP on Lotus `Limited 45`. All traces are the mean PSNR of 5 runs (varying the seed). The notations ▲ and ⋆ denote the initial PSNR and rise time, respectively, and the horizontal dashed line indicates steady PSNR of DIP (noise).


EDIP (FBP)

DIP (noise)

Reference

PSNR: 29.66 dB, SSIM: 0.8330

PSNR: 29.77 dB, SSIM: 0.8353

EDIP (FBP) initial

DIP (noise) initial

FBP

PSNR: 27.55 dB, SSIM: 0.7278

PSNR: 17.22 dB, SSIM: 0.3798

Fig. 19. Lotus reconstruction of EDIP versus DIP on Lotus `Limited 45` data. From the 5 runs (varying the seed), the one with the (closest to) median PSNR was selected for each method. The reported reconstructions are the best reconstruction (i.e. reconstruction at the minimum loss value).

B. The Walnut (Continued)

The quantitative results in Table IX validate our findings on the Lotus root. See also Figs. 21 and 22–23 for convergence behavior and exemplary reconstructions.

TABLE IX

Quantitative evaluation for Walnut Sparse 120 with EDIP being pretrained on ellipses data. For the experiments marked with "*" a higher initial learning rate was used (see Table V).

Ellipses-Walnut Sparse 120	Rise time	(Max PSNR; iters)	Steady PSNR	Init PSNR
DIP (noise)	20 373	(34.02; 25 357)	33.87	6.88
DIP (FBP)	13 778	(34.07; 28 094)	33.90	6.26
DIP-FE (noise)	14 289	(34.02; 23 573)	33.88	6.88
DIP-FE (FBP)	13 421	(34.19; 23 266)	33.97	6.26
EDIP (FBP)	4496	(33.92; 13 039)	33.56	25.67
EDIP (noise) *	9561	(34.12; 23 352)	33.95	12.22
EDIP-FE (FBP)	4384	(33.91; 12 540)	33.70	25.67
EDIP-FE (noise) *	21 760	(33.89; 29 159)	33.75	12.22
TV	–	–	31.67	–

TABLE X

Quantitative evaluation for Walnut 3D Sparse 20 and 3D Sparse 60 with EDIP being pretrained on ellipsoids data. Both maximum PSNR and steady PSNR are computed using the iteration-wise median PSNR history over 3 repeated runs (varying the random seed). For steady PSNR, the median value of the median PSNR history over the last 5k iterations is considered. The convergence of TV is very stable, and we report the final PSNR. Initial PSNR is the mean value over the 3 repeated runs. All PSNR values are in dB.

Ellipsoids-Walnut	3D Sparse 20				3D Sparse 60			
	Rise time	(Max PSNR; iters)	Steady PSNR	Init PSNR	Rise time	(Max PSNR; iters)	Steady PSNR	Init PSNR
DIP (noise)	17 200	(30.68; 23 477)	30.37	7.29	49 041	(34.05; 58 901)	33.93	7.29
DIP (FBP)	13 016	(31.32; 25 063)	31.19	8.19	27 873	(34.37; 53 731)	34.22	8.62
EDIP (FBP)	3739	(31.48; 10 689)	30.94	19.77	11 247	(34.35; 40 810)	34.18	20.17
EDIP-FE (FBP)	2979	(31.38; 10 749)	30.93	19.77	14 520	(34.33; 45 259)	34.15	20.17
TV	–	–	28.89	–	–	–	33.35	–

Supplementary Material D
Validating Pretraining

Different checkpoints are obtained from multiple pretraining runs (varying the random seed), and by collecting checkpoints along the optimization trajectory from each run. We identify the parameters' configuration to be used at test time from these checkpoints by selecting the one with the best performance on a validation set. To this end, we design a reconstructive task based on the Shepp-Logan phantom, a standard test image created to assess reconstruction algorithms. The phantom is by construction within the ellipses data manifold and shares the same noise distribution of ellipses measurements. The checkpoint leading to the shortest rise time is selected, among those with a steady PSNR that is at most 0.25 dB lower than the maximum reached steady PSNR.

We repeat the pretraining three times (varying the seed) and collect checkpoints after every 20 epochs for Lotus Sparse 20 and Lotus Limited 45, training for a maximum of 100 epochs. We also include the checkpoint for which the model shows minimum validation loss. For Walnut Sparse 120 we pretrain for 20 epochs, and retain only the minimum validation loss checkpoint. Fig. 24 shows the convergence of the pretraining on the ellipses datasets for the Lotus and the Walnut settings, along with the learning rate scheduling.

At the validation stage, each checkpoint is evaluated by performing EDIP fine-tuning on simulated data of the Shepp-Logan phantom. The validation runs for Lotus Sparse 20, Lotus Limited 45, and Walnut Sparse 120 are shown in Figs. 25 and 27, respectively. In the Lotus settings, starting EDIP fine-tuning using checkpoints from a later epoch (e.g. 60, 80, 100) is more beneficial. Nonetheless, even pretraining for fewer epochs (e.g. 20) can already greatly benefit the EDIP fine-tuning, although to a lesser degree. Pretraining considerably ameliorates the quality of the reconstruction of the Shepp-Logan phantom for both Lotus and Walnut settings. Especially for the Lotus Limited 45 setting, it substantially increases the reconstruction quality.

We then investigate whether the selected checkpoints that then are used for the test data — both the Lotus and the Walnut could be considered an out-of-distribution image class — are still optimal as we switch from the simulated measurements of the Shepp-Logan phantom to the real-measured test data. Fig. 26 and Fig. 28 show the PSNR convergence on the test data using different checkpoints. While we observe a different behavior between validation and test data, the validation selects one of the best two checkpoint.

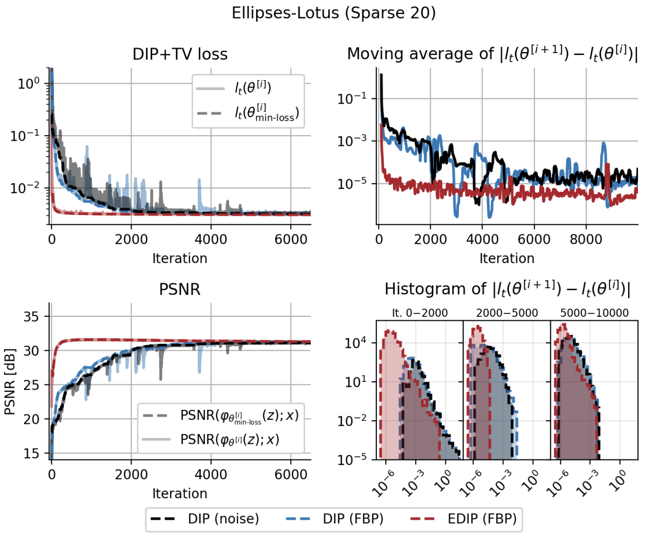


Fig. 20. Min-loss and PSNR computed with the min-loss output for Lotus `Sparse 20` (left). Loss variation (i.e. $|l_t(\theta^{[i+1]}) - l_t(\theta^{[i]})|$) and respective histograms computed over three intervals (right). The moving average uses a window size of 100 iterations.

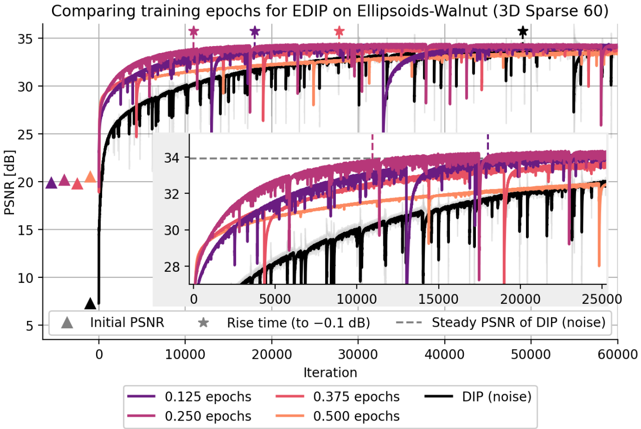


Fig. 21. The optimization of EDIP using different checkpoints for EDIP (FBP) on Walnut `3D Sparse 60` data. The notations ▲ and ⋆ denote the initial PSNR and rise time, respectively, and the horizontal dashed line indicates steady PSNR of DIP (noise).

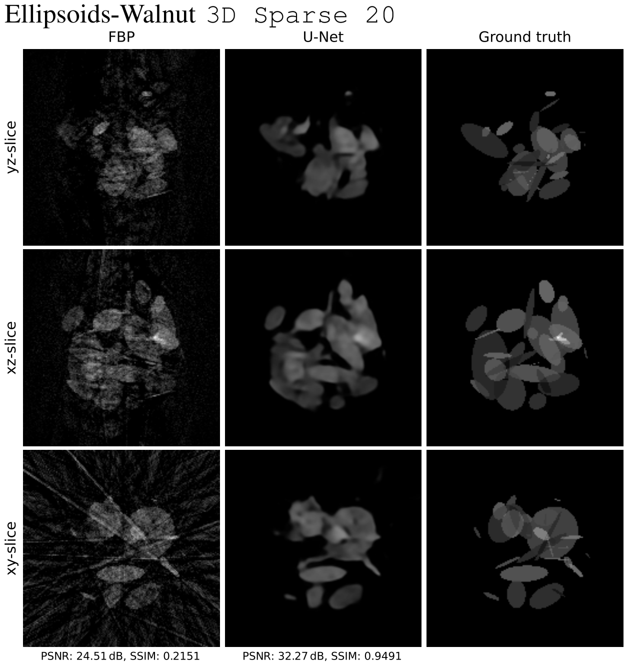

Ellipsoids-Walnut 3D Sparse 20
FBP
U-Net
Ground truth
yz-slice
xz-slice
xy-slice
PSNR: 24.51 dB, SSIM: 0.2151
PSNR: 32.27 dB, SSIM: 0.9491

Fig. 22. Exemplary reconstructions from the synthetic training dataset of ellipsoids images for Walnut `3D Sparse 20`.

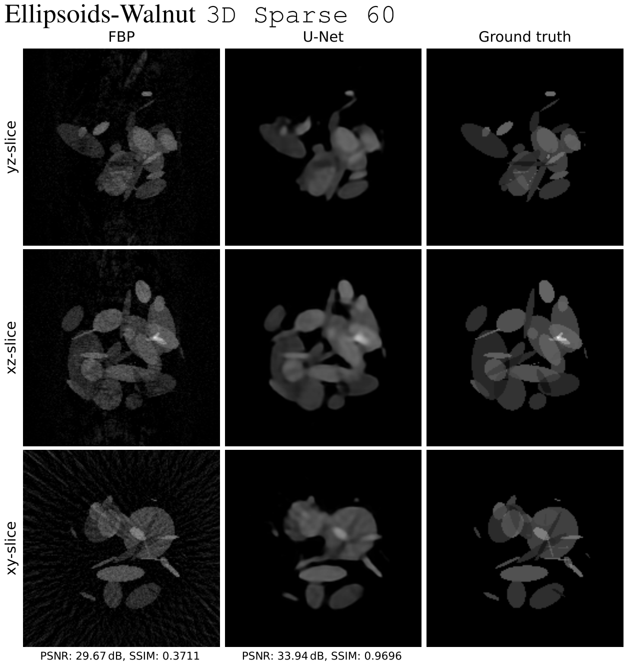

Ellipsoids-Walnut 3D Sparse 60
FBP
U-Net
Ground truth
yz-slice
xz-slice
xy-slice
PSNR: 29.67 dB, SSIM: 0.3711
PSNR: 33.94 dB, SSIM: 0.9696

Fig. 23. Exemplary reconstructions from the synthetic training dataset of ellipsoids images for Walnut `3D Sparse 60`.

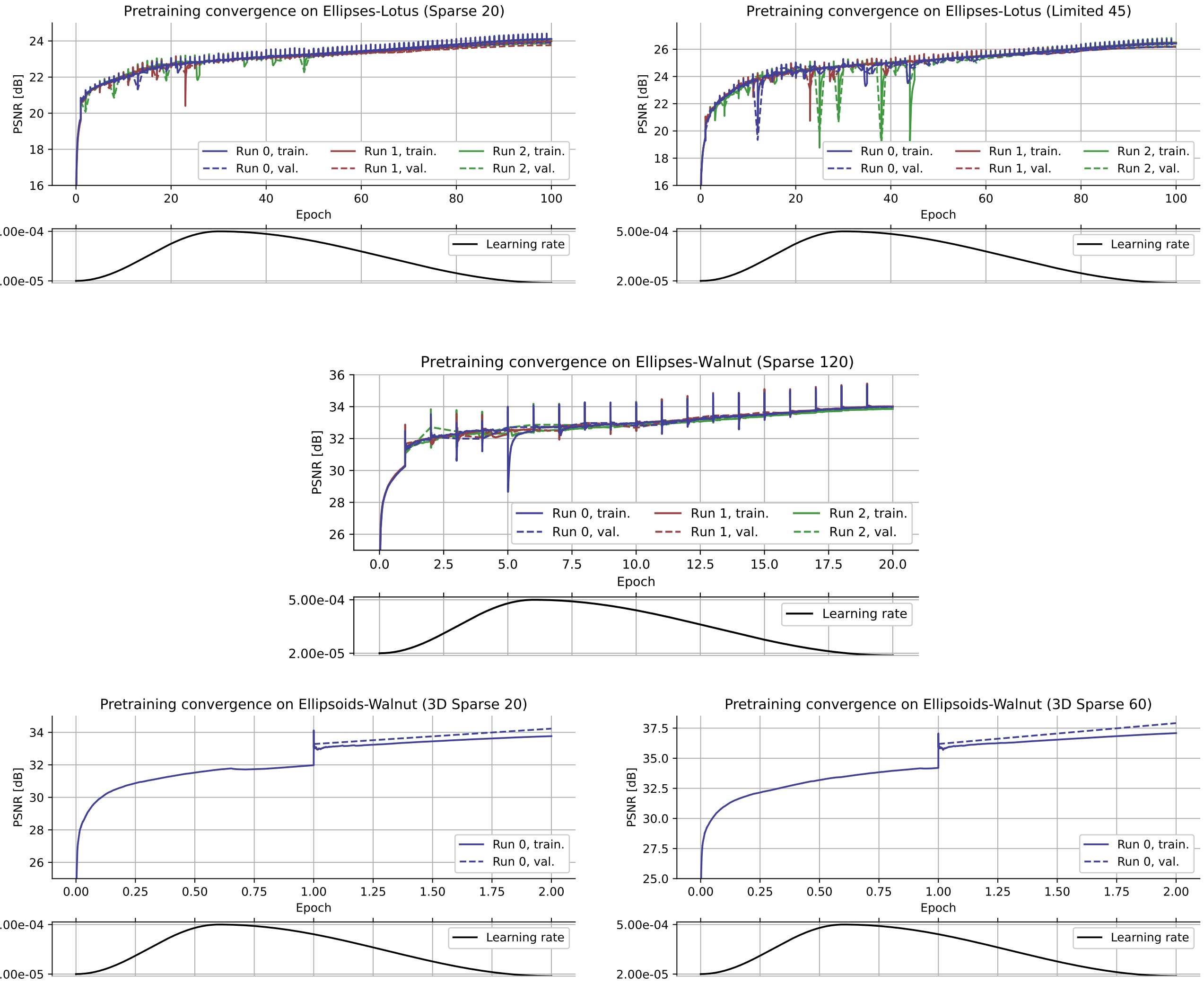


Fig. 24. Pretraining convergence. Solid lines show the running mean of the training loss since the start of the respective epoch; dashed lines show the mean validation loss evaluated after each epoch (on a set of 3200 held-out images).

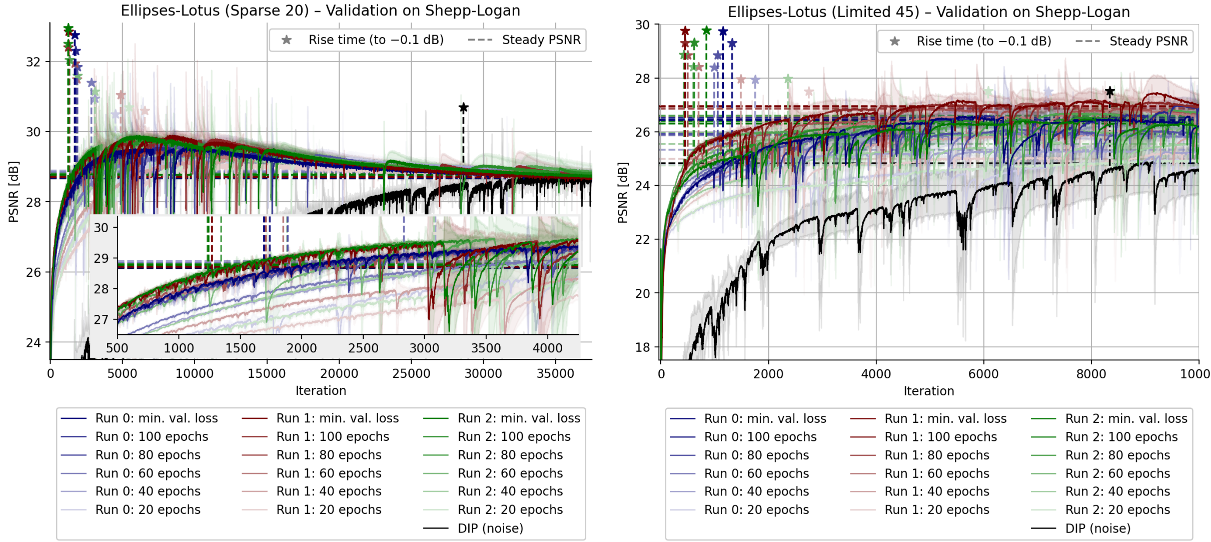


Fig. 25. Validation runs on the Shepp-Logan phantom for selecting the initial EDIP (FBP) model parameters for data in the Lotus `Sparse 20` and `Limited 45` geometry. For `Sparse 20` the model from training run 2 after 100 epochs is selected because it has the shortest rise time (with a sufficiently high steady PSNR), whilst, for `Limited 45` run 1 after 100 epochs is selected.

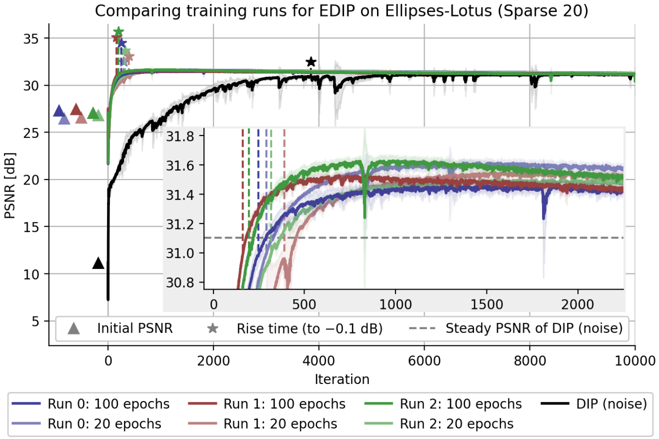


Fig. 26. Optimization of EDIP using different checkpoints considered during validation (see Fig. 25) for EDIP (FBP) on Lotus `Sparse 20` data. The parameters from run 2 after 100 epochs are selected by the validation. The notations ▲ and ⋆ denote the initial PSNR and rise time, respectively, and the horizontal dashed line indicates steady PSNR of DIP (noise).

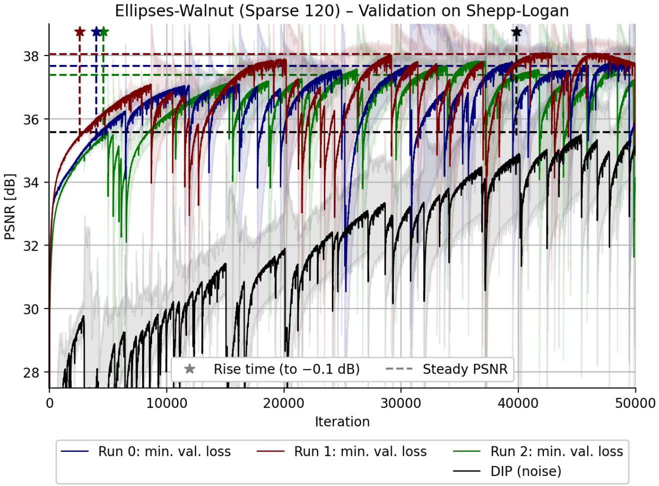


Fig. 27. Validation runs on the Shepp-Logan phantom for selecting the initial EDIP (FBP) model parameters for the Walnut `Sparse 120` geometry. The model from training run 1 is selected because it has the shortest rise time (with a high steady PSNR).

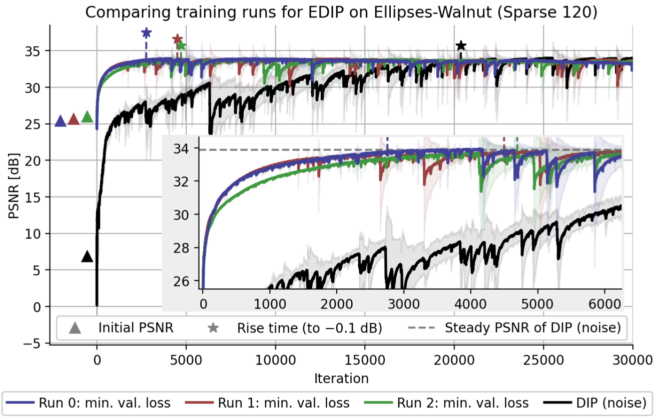


Fig. 28. Optimization of EDIP using parameters from different training runs considered during validation (see Fig. 27) for EDIP (FBP), on Walnut `Sparse 120` data. The parameters from run 1 are selected by the validation. The notations ▲ and ⋆ denote the initial PSNR and rise time, respectively, and the horizontal dashed line indicates steady PSNR of DIP (noise).

Supplementary Material E
Ablation Study and Limitations

We showcase one potential pitfall of the “supervised pretraining + unsupervised fine-tuning” paradigm for DIP, resorting to a by far too specific and less diverse image class. Instead of the ellipses dataset, we use human brain images for the supervised learning stage. We consider MRI images of the human brain from the ACRIN-FMISO-Brain (ACRIN 6684) dataset from https://wiki.cancerimagingarchive.net/x/kQIGAg. For the synthetic dataset, we normalize the extracted 2D slices and (mis)interpret the values to be X-ray attenuation coefficients. We use a random data split on patient level, leading to 30 917 training images and 4524 validation images. Both training and validation images are augmented by random rotations. Fig. 29 shows an exemplary reconstruction of the brain dataset, whilst Fig. 30 reports the pretraining convergence.

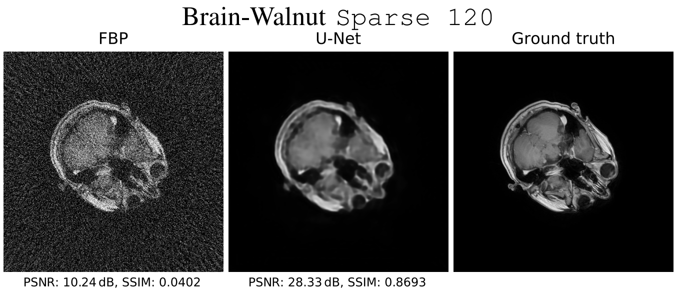


Fig. 29. Exemplary reconstructions from the synthetic training dataset of brain images for Walnut `Sparse 120`.

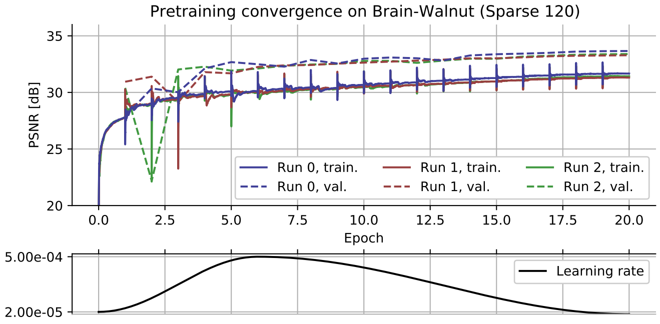


Fig. 30. Pretraining convergence.

In Fig. 31, we show the validation on the Shepp-Logan. Fig. 32 compares DIP and EDIP trained on the brain dataset. EDIP performs worse than DIP.

Fig. 33 suggests that checkpoints from repeated pretraining runs also lead to similar subpar results. We observe the inadequacy of the brain dataset (of its education!). Pretraining on the brain dataset induces too dataset specific inductive biases from which EDIP fails to escape, leading to slow convergence and sub-optimal steady PSNR. Possibly the implicit regularization exerted by the pretraining on the brain dataset essentially restricts the networks from leaving a “pretrained landscape” of sub-optimal parameters’ configurations.

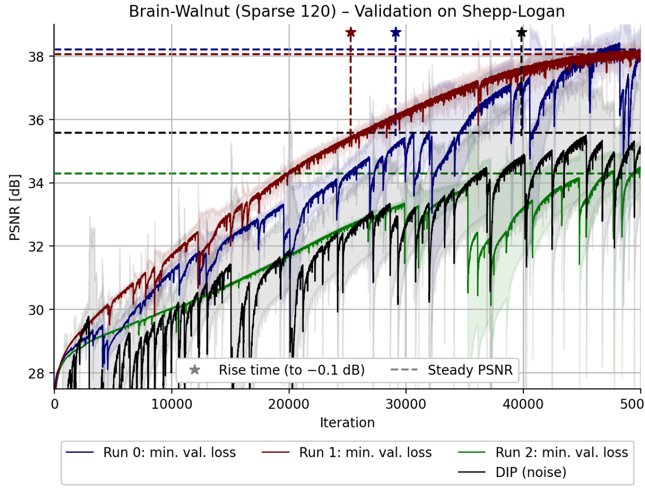


Fig. 31. Validation runs on the Shepp-Logan phantom for selecting the initial EDIP (FBP) model parameters pretrained on the brain dataset for data in the Walnut `Sparse 120` geometry. The model from training run 1 is selected because it has the shortest rise time. Despite the relatively high number of 50k iterations, the (E)DIP optimizations do not fully converge yet.

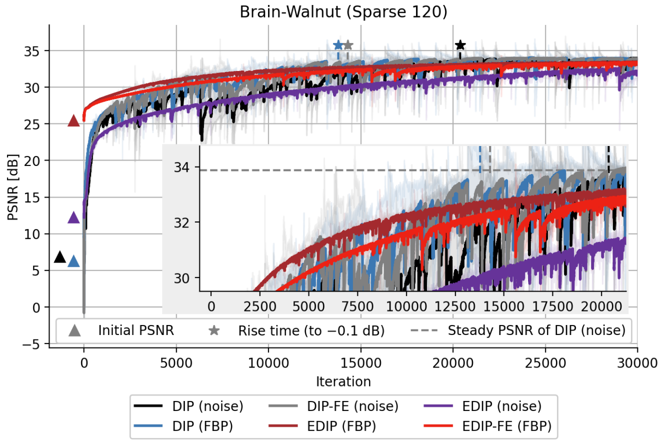


Fig. 32. The optimization of EDIP versus DIP pretrained on the brain dataset compared to standard DIP on Walnut `Sparse 120` measurement data. All traces are the mean PSNR of 5 repetitions of the same experimental run (varying the random seed). See Tab. XI for complementary tabular results. The notations ▲ and ⋆ denote the initial PSNR and rise time, respectively.

TABLE XI
Quantitative evaluation results for EDIP on Walnut `Sparse 120` after pretraining on the brain dataset for 20 epochs. No rise time can be reported, because the PSNR is not reaching the steady PSNR of DIP (noise) minus 0.1 dB within the 30k iterations. See Table IX for the corresponding results from standard DIP and from pretraining on ellipses data.

Brain-Walnut `Sparse 120` — pretrained for 20 epochs

	Rise time	(Max PSNR; iters)	Steady PSNR	Init PSNR
EDIP (FBP)	–	(33.51; 29 982)	33.35	25.49
EDIP (noise)	–	(32.67; 29 875)	32.29	12.23
EDIP-FE (FBP)	–	(33.43; 29 862)	33.24	25.49
EDIP-FE (noise)	–	(31.06; 29 989)	30.39	12.23

We then check whether using earlier checkpoints would lead to better transferable performances. We, indeed, observe that an early-stopping of the pretraining stage on the brain dataset ameliorates EDIP, cf. Fig. 35. The longer we pretrain on the brain dataset, the worse EDIP performs subsequently.

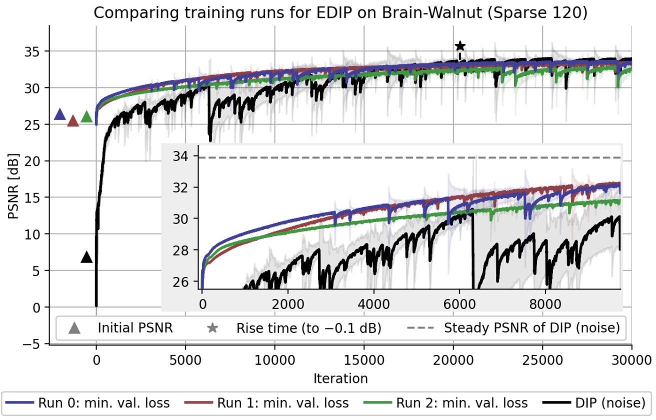
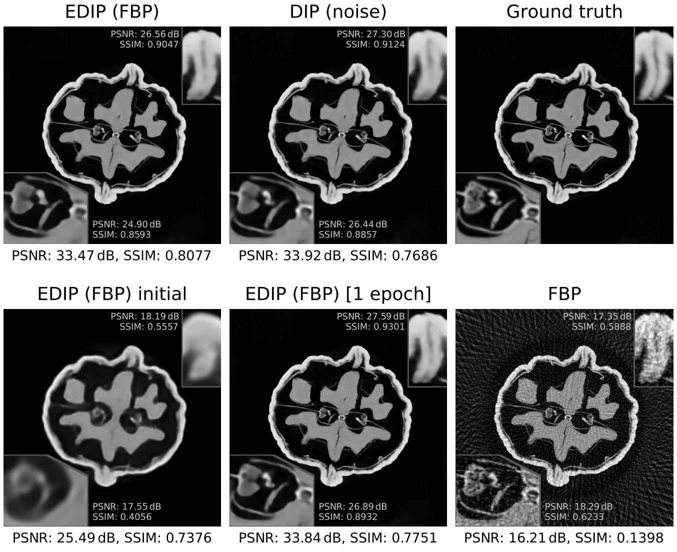


Fig. 33. The optimization of EDIP using parameters from different training runs considered during validation (see Fig. 31) for EDIP (FBP), pretrained on the brain dataset, on Walnut `Sparse 120` measurement data. The parameters from run 1 are the ones selected by the validation. The notations ▲ and ⋆ denote the initial PSNR and rise time, respectively, and the horizontal dashed line indicates steady PSNR of DIP (noise).


EDIP (FBP) | DIP (noise) | Ground truth

PSNR: 33.47 dB, SSIM: 0.8077 | PSNR: 33.92 dB, SSIM: 0.7686

EDIP (FBP) initial | EDIP (FBP) [1 epoch] | FBP

PSNR: 25.49 dB, SSIM: 0.7376 | PSNR: 33.84 dB, SSIM: 0.7751 | PSNR: 16.21 dB, SSIM: 0.1398

Fig. 34. Walnut reconstruction of EDIP pretrained on brain dataset, compared to standard DIP. From the 5 runs (varying the seed), the one with the (closest to) median PSNR was selected for each method. See Fig. 8 for the Walnut reconstruction with EDIP pretrained on the ellipses dataset.

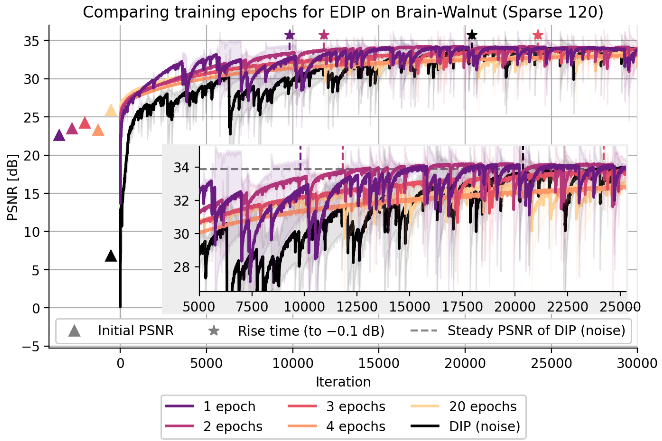


Fig. 35. The optimization of EDIP using parameters from different epochs for EDIP (FBP) on Walnut `Sparse 120` measurement data while pretraining on the brain dataset. The notations ▲ and ⋆ denote the initial PSNR and rise time, respectively, and the horizontal dashed line indicates steady PSNR of DIP (noise).