Journal of Mathematical Imaging and Vision
https://doi.org/10.1007/s10851-019-00901-3

Networks for Nonlinear Diffusion Problems in Imaging

S. Arridge[1] · **A. Hauptmann**[1,2]

Received: 29 November 2018 / Accepted: 23 August 2019

Abstract
A multitude of imaging and vision tasks have seen recently a major transformation by deep learning methods and in particular by the application of convolutional neural networks. These methods achieve impressive results, even for applications where it is not apparent that convolutions are suited to capture the underlying physics. In this work, we develop a network architecture based on nonlinear diffusion processes, named *DiffNet*. By design, we obtain a nonlinear network architecture that is well suited for diffusion-related problems in imaging. Furthermore, the performed updates are explicit, by which we obtain better interpretability and generalisability compared to classical convolutional neural network architectures. The performance of DiffNet is tested on the inverse problem of nonlinear diffusion with the Perona–Malik filter on the STL-10 image dataset. We obtain competitive results to the established U-Net architecture, with a fraction of parameters and necessary training data.

Keywords Neural networks · Deep learning · Partial differential equations · Nonlinear diffusion · Image flow · Nonlinear inverse problems

1 Introduction

We are currently undergoing a paradigm shift in imaging and vision tasks from classical analytic to learning and data-based methods. In particular, this shift is driven by deep learning and the application of convolutional neural networks (CNN). Whereas highly superior results are obtained, interpretability and analysis of the involved processes is a challenging and ongoing task [20,24].

In a general setting, deep learning proposes to develop a nonlinear mapping $A_\Theta : X \to Y$ between elements of two spaces X, Y (which may be the same) parametrised by a finite set of parameters Θ, which need to be learned:

$$g = A_\Theta f \tag{1.1}$$

This learning-based approach is in marked contrast to classical methods with a physical interpretation of the process (1.1) for both computational modelling of data given a physical model (which we call a *forward problem*) and the estimation of parameters of a physical model from measured, usually noisy, data (which we call an *inverse problem*). Several recent papers have discussed the application of deep learning in both forward [23,31,36,37,40,41] and inverse [21,22,29,42–44] problems. Several questions become essential when applying learned models in such cases including:

- how and to what extent can learned models replace physical models?
- how do learned models depend on training protocols and how well do they generalise?
- what are appropriate architectures for the learned models, what is the size of the parameter set Θ that needs to be learned, and how can these be interpreted?

In this paper, we aim to answer some of these questions, by taking a few steps back and looking at an analytic motivation for network architectures. Here, we consider in

This work was partially supported by the Academy of Finland (Project 312123, Finnish Centre of Excellence in Inverse Modelling and Imaging, 2018–2025) and by British Heart Foundation Grant NH/18/1/33511, EPSRC Grant EP/M020533/1, and EPSRC-Wellcome Grant WT1019572. We acknowledge the support of NVIDIA Corporation with one Titan Xp GPU. Codes will be made available on http://www.ImageFlowNet.org/.

✉ A. Hauptmann
andreas.hauptmann@oulu.fi

S. Arridge
s.arridge@ucl.ac.uk

[1] Department of Computer Science, University College London, London, UK

[2] Research Unit of Mathematical Sciences, University of Oulu, Oulu, Finland

Published online: 13 September 2019

particular mappings between images u in dimension d, i.e. $X = Y = L^p(\Omega \subset \mathbb{R}^d)$, for which there exist several widely used linear and nonlinear mappings A defined by differential and/or integral operators. For example, a general linear integral transform such as

$$u^{\text{obs}}(x) = (Au^{\text{true}})(x) = \int_\Omega K(x, y)u^{\text{true}}(y)\mathrm{d}y \tag{1.2}$$

includes stationary convolution as a special case if the kernel is translation invariant, i.e. $K(x, y) \equiv K(x-y)$. If the kernel depends on the image, i.e. $K(x, y) \equiv K(x, y; u)$, then (1.2) becomes nonlinear. Alternatively, the forward mapping may be modelled as the end-point of an evolution process which becomes nonlinear if the kernel depends on the image state $K(x, y, t) \equiv K(x, y; u(t))$; see Eq. (2.7) below for a specific example.

Furthermore, a widely studied class of image mappings is characterised by defining the evolution through a partial differential equation (PDE) where flow is generated by the local structure of u [25,34,38], such that

$$u_t = F\left(u, \nabla u, \frac{\partial^2 u}{\partial x_i \partial x_j}, \dots\right). \tag{1.3}$$

These problems in general do not admit an explicit integral transform representation and are solved instead by numerical techniques, but since they depend on a physical model, described by the underlying PDE, they can be analysed and understood thoroughly [39]. This also includes a statistical interpretation [19] as hyperpriors in a Bayesian setting [5]. Furthermore, models like (1.3) can be used to develop priors for nonlinear inverse problems, such as optical tomography [8,15] and electrical impedance tomography [13].

Motivated by the success of these analytic methods to imaging problems in the past, we propose to combine physical models with data-driven methods to formulate network architectures for solving both forward and inverse problems that take the underlying physics into account. We limit ourselves to the case where the physical model is of diffusion type, although more general models could be considered in the future. The leading incentive is given by the observation that the underlying processes in a neural network do not need to be limited to convolutions.

Similar ideas of combining partial differential equations with deep learning have been considered earlier. For instance, this is done by learning of a PDE via optimal control [26], as well as deriving CNN architectures motivated by diffusion processes [6], deriving stable architectures by drawing connections to ordinary differential equations [11] and constraining CNNs [33] by the interpretation as a partial differential equation. 'PDE-NET 2.0' [27,28] is a recent example of a network architecture designed to learn dynamic PDEs assumed to be of the form (1.3) where the function F is learned as a polynomial in convolution filters with appropriate vanishing moments. Another interpretation of our approach can be seen as introducing the imaging model into the network architecture; such approaches have led to a major improvement in reconstruction quality for tomographic problems [2,14,17].

This paper is structured as follows. In Sect. 2, we review some theoretical aspects of diffusion processes for imaging and the inversion based on theory of partial differential equations and differential operators. We formulate the underlying conjecture for our network architecture that the diffusion process can be inverted by a set of local non-stationary filters. In the following, we introduce the notion of *continuum networks* in Sect. 3 and formally define the underlying *layer operator* needed to formulate network architectures in a continuum setting. We draw connections to the established convolutional neural networks in our continuum setting. We then proceed to define the proposed layer operators for diffusion networks in Sect. 4 and derive an implementable architecture by discretising the involved differential operator. In particular, we derive a network architecture that is capable of reproducing inverse filtering with regularisation for the inversion of nonlinear diffusion processes. We examine the reconstruction quality of the proposed *DiffNet* in following Sect. 5 for an illustrative example of deconvolution and the challenging inverse problem of inverting nonlinear diffusion with the Perona–Malik filter. We achieve results that are competitive to popular CNN architectures with a fraction of the amount of parameters and training data. Furthermore, all computed components that are involved in the update process are interpretable and can be analysed empirically. In Sect. 6, we examine the generalisablity of the proposed network with respect to necessary training data. Additionally, we empirically analyse the obtained filters and test our underlying conjecture. Section 7 presents some conclusions and further ideas.

2 Diffusion and Flow Processes for Imaging

In the following, we want to explore the possibility to include a nonlinear process as an underlying model for network architectures. Specifically, the motivation for this study is given by diffusion processes that have been widely used in imaging and vision. Let us consider here the general diffusion process in $\mathbb{R}^d$, then on a fixed time interval with some diffusivity γ, to be defined later, we have

$$\begin{cases} \partial_t u &= \nabla \cdot (\gamma \nabla u) \text{ in } \mathbb{R}^d \times (0, T] \\ u(x, 0) &= u_0(x) \quad\quad \text{in } \mathbb{R}^d. \end{cases} \tag{2.1}$$

Remark 1 When considering bounded domains $\Omega \subset \mathbb{R}^d$, we will augment (2.1) with boundary conditions on $\partial\Omega$. We return to this point in Sect. 4.

In the following, we denote the spatial derivative by

$$(\mathcal{L}(\gamma)u)(x,t) := \nabla \cdot (\gamma \nabla u(x,t)). \tag{2.2}$$

Let us first consider the *isotropic diffusion* case, then the differential operator becomes the spatial Laplacian $\mathcal{L}(\gamma = 1) = \Delta$. In this case, the solution of (2.1) at time T is given by convolution with a *Green's function*

$$u_T(x) = G_{\sqrt{2T}}(x) * u_0(x) \tag{2.3}$$

where $G_{\sqrt{2T}} = \frac{1}{(4\pi T)^{d/2}} \exp\left[-\frac{x^2}{4T}\right]$ in dimension d and we recall that the convolution of two functions $f, g \in L^1(\mathbb{R}^d)$ is defined by

$$(g * f)(x) = \int_{\mathbb{R}^d} g(x-y) f(y) \mathrm{d}y. \tag{2.4}$$

Definition 1 The *Green's operator* is defined with the Green's function as its kernel

$$\mathcal{G}_{\frac{1}{2}\sigma^2} u := \int_{\mathbb{R}^d} \frac{u(x)}{(2\pi\sigma^2)^{d/2}} \exp\left[-\frac{|x-y|^2}{2\sigma^2}\right] \mathrm{d}y \tag{2.5}$$

by which we have

$$u_T(x) = \mathcal{G}_T u_0(x)$$

In the general case for an *anisotropic diffusion flow* (ADF), we are interested in a scalar *diffusivity* $\gamma \in [0,1]$ that depends on u itself, i.e.

$$\partial_t u = \nabla \cdot (\gamma(u) \nabla u) \tag{2.6}$$

This is now an example of a nonlinear evolution

$$u_T(x) = \mathcal{K}_T u_0 = \int_0^T \int_{\mathbb{R}^d} K^{\mathrm{ADF}}(x, y, u(y,t)) u_0(y) \mathrm{d}y \mathrm{d}t \tag{2.7}$$

where $K^{\mathrm{ADF}}(x, y, u(x,t))$ is now a non-stationary, nonlinear and time-dependent kernel. In general, there is no explicit expression for K^{ADF} and numerical methods are required for the solution of (2.6).

Remark 2 Not considered here, but a possible extension to (2.6) is where γ is a tensor, which for $d = 2$ takes the form

$$\partial_t u = \nabla \cdot \begin{pmatrix} \gamma_{11} & \gamma_{12} \\ \gamma_{12} & \gamma_{22} \end{pmatrix} \nabla u$$

Furthermore, extensions exist for the case where u is vector or tensor valued. We do not consider these cases here, see [38] for an overview.

2.1 Forward Solvers

First of all, let us establish a process between two states of the function u. Integrating over time from $t = t_0$ to $t = t_1 = t_0 + \delta t$ yields

$$\int_{t_0}^{t_1} \partial_t u(x,t) \, \mathrm{d}t = \int_{t_0}^{t_1} (\mathcal{L}(\gamma)u)(x,t) \, \mathrm{d}t.$$

Note that the left-hand side can be expressed as $\int_{t_0}^{t_1} \partial_t u(x,t) \, \mathrm{d}t = u(x,t_1) - u(x,t_0)$ and we denote the right-hand side by an integral operator $\mathcal{A}_{\delta t}(\gamma)$, such that

$$(\mathcal{A}_{\delta t}(\gamma)u)(x,t_0) := \int_{t_0}^{t_1 = t_0 + \delta t} (\mathcal{L}(\gamma)u)(x,t) \, \mathrm{d}t. \tag{2.8}$$

In the following, we denote the solution of (2.1) at time instances t_n as $u^{(n)} = u(x,t_n)$. Then, we can establish a relation between two time instances of u by

$$u^{(n+1)} = (\mathrm{Id} + \mathcal{A}_{\delta t}(\gamma)) u^{(n)} = u^{(n)} + \int_{t_n}^{t_{n+1}} \mathcal{L}(\gamma) u(x,t) \, \mathrm{d}t, \tag{2.9}$$

where Id denotes the identity and $t_{n+1} = t_n + \delta t$.

Since we cannot compute $u^{(n+1)}$ by (2.9) without the explicit knowledge of the (possibly time dependent) diffusivity γ, it is helpful to rather consider a fixed diffusivity at each time instance $\gamma^{(n)} = \gamma(x, t = t_n)$, or $\gamma^{(n)} = \gamma(u(x, t = t_n))$ in the nonlinear case; then, by using the differential operator (2.2), we have an approximation of (2.8) by

$$\delta t \mathcal{L}(\gamma^{(n)}) u^{(n)} = \delta t (\nabla \cdot \gamma^{(n)} \nabla u^{(n)}) \approx \mathcal{A}_{\delta t}(\gamma) u^{(n)}.$$

We can now solve (2.6) approximately by iterating for time steps δt using either an *explicit* scheme

$$\mathcal{D}_{\delta t}^{\mathrm{Expl}}(\gamma^{(n)}) u^{(n)} = \left(\mathrm{Id} + \delta t \mathcal{L}(\gamma^{(n)})\right) u^{(n)}, \tag{2.10}$$

or an *implicit* scheme

$$\mathcal{D}_{\delta t}^{\mathrm{Impl}}(\gamma^{(n)}) u^{(n)} = \left(\mathrm{Id} - \delta t \mathcal{L}(\gamma^{(n)})\right)^{-1} u^{(n)}, \tag{2.11}$$

whereas (2.10) is stable only if CFL conditions are satisfied and (2.11) is unconditionally stable, they are both only accurate for sufficiently small steps δt. In fact, by the Neumann series, the schemes are equivalent for small δt as

$$(\mathrm{Id} - \delta t \mathcal{L}(\gamma))^{-1} = \mathrm{Id} + \delta t \mathcal{L}(\gamma) + \mathcal{O}((\delta t)^2) \tag{2.12}$$

and coincide with the integral formulation of (2.9).

It is also useful to look at the Green's function's solutions.

Lemma 1 *Consider the isotropic case with $\gamma \equiv 1$. Then, we may write, with time steps $\delta t = T/N$,*

$$\begin{aligned}\mathcal{G}_T u_0 &= G_{\sqrt{2T}} * u_0 \\ &= \underbrace{G_{\sqrt{2\delta t}} * \ldots * G_{\sqrt{2\delta t}}}_{N\text{-times}} * u_0 = \underbrace{\mathcal{G}_{\delta t} \circ \ldots \mathcal{G}_{\delta t}}_{N\text{-times}} \circ u_0\end{aligned} \tag{2.13}$$

Proof Take the Fourier Transform[1]

$$\hat{G}_\sigma(k) = \mathcal{F}_{x\to k} G_\sigma(x) = \mathrm{e}^{-\frac{\sigma^2 k^2}{2}}$$

and use the convolution theorem to give

$$\begin{aligned}\hat{G}_{\sqrt{2T}}(k)\hat{u}_0(k) &= \left(\Pi_{n=1}^N \hat{G}_{\sqrt{2\delta t}}(k)\right)\hat{u}_0(k) \\ \mathrm{e}^{-k^2 T}\hat{u}_0(k) &= \left(\mathrm{e}^{-k^2\delta t}\right)^N \hat{u}_0(k) = \mathrm{e}^{-k^2 N\delta t}\hat{u}_0(k),\end{aligned}$$

which gives the claim. □

Let us also note that in Fourier domain, by Taylor series expansion, we have

$$\exp(-k^2\delta t) \to 1 - k^2\delta t + \frac{1}{2}k^4(\delta t)^2 - \cdots,$$

and therefore, in the spatial domain, the finite difference step and the Gaussian convolution step are the same

$$\begin{aligned}\lim_{\delta t\to 0}\left(G_{\sqrt{2\delta t}} * u_0\right) &= (\mathrm{Id} + \delta t\Delta) * u_0 \\ &= \lim_{\delta t\to 0}(\mathrm{Id} - \delta t\Delta)^{-1} u_0\,.\end{aligned}$$

2.2 Inverse Filtering

Let us now consider the inverse problem of reversing the diffusion process. That is we have u_T and aim to recover the initial condition u_0. This is a typical ill-posed problem as we discuss in the following.

2.2.1 Isotropic Case $\gamma \equiv 1$

As the forward problem is represented as convolution in the spatial domain, the inverse mapping $u_T \mapsto u_0$ is a (stationary) *deconvolution*. We remind that $\hat{u}_T = \mathrm{e}^{-k^2 T}\hat{u}_0(k)$; then, the inversion is formally given by division in the Fourier domain as

$$u_0(x) = \mathcal{F}^{-1}_{k\to x}\left[\hat{u}_T(k)\mathrm{e}^{k^2 T}\right]. \tag{2.14}$$

However, we note:

(i) The factor $\mathrm{e}^{k^2 T}$ is unbounded, and hence, the equivalent convolution kernel in the spatial domain does not exist.
(ii) Equation (2.14) is unstable in the presence of even a small amount of additive noise, and hence, it has to be *regularised* in practice.

Nevertheless, let us consider formally with $\mathrm{e}^{k^2 T} = \left(\mathrm{e}^{k^2\delta t}\right)^N$ that by Taylor series, we get

$$\begin{aligned}\mathcal{F}^{-1}_{k\to x}\mathrm{e}^{k^2\delta t} &\approx \mathcal{F}^{-1}_{k\to x}\left[1 + k^2\delta t + \frac{1}{2}(k^2\delta t)^2 + \cdots\right] \\ &= 1 - \delta t\Delta + \mathcal{O}((\delta t)^2)\,.\end{aligned}$$

Motivated by this, we define an operator for the inversion process

$$\mathcal{E}^{\mathrm{iso}}_{\delta t} u := (\mathrm{Id} - \delta t\Delta)\, u \simeq \mathcal{G}^{-1}_{\delta t} u. \tag{2.15}$$

Clearly, $\mathcal{E}^{\mathrm{iso}}_{\delta t}$ coincides with the inverse of the implicit update in (2.11), and

$$\tilde{u}_0 = \underbrace{\mathcal{E}^{\mathrm{iso}}_{\delta t} \circ \ldots \circ \mathcal{E}^{\mathrm{iso}}_{\delta t}}_{N\text{-times}} \circ u_T \tag{2.16}$$

is an estimate for the deconvolution problem which (in the absence of noise) is correct in the limit

$$\lim_{\delta t\to 0}\tilde{u}_0 \to u_0. \tag{2.17}$$

2.2.2 Anisotropic Case

In this case, the diffused function is given by (2.7). Following Lemma 1, we may put

$$u_T = \mathcal{K}_T u_0 \simeq \tilde{u}_T := \mathcal{D}^{\mathrm{Expl}}_{\delta t}(\gamma^{(N-1)}) \circ \cdots \circ \mathcal{D}^{\mathrm{Expl}}_{\delta t}(\gamma^{(0)}) u_0 \tag{2.18}$$

and we also have

$$\lim_{\delta t\to 0}\tilde{u}_T \to u_T. \tag{2.19}$$

[1] Note the definition of Fourier Transform is chosen to give the correct normalisation so that $\hat{u}(k)|_{k=0} = \int_{\mathrm{R}^d} u(x)\mathrm{d}^n x$.

Conjecture 1 There exists a set of local (non-stationary) filters $\mathcal{E}_{\delta t}(\zeta)$ where

$$\mathcal{E}_{\delta t}(\zeta)u = u - \delta t \int_{\mathbb{R}^d} \zeta(x, y)u(y)\mathrm{d}y \tag{2.20}$$

and where $\zeta(x, y)$ has only local support and such that

$$u_0 = \mathcal{K}_T^{-1} u_T \quad \simeq \quad \tilde{u}_0 := \mathcal{E}_{\delta t}(\zeta^{(N-1)}) \circ \cdots \circ \mathcal{E}_{\delta t}(\zeta^{(0)}) u_T \,. \tag{2.21}$$

Remark 3 (Unsharp Masking) We recall that a simple method for "deconvolution" is called *Unsharp Masking* which is usually considered as

$$u^{\mathrm{obs}} \mapsto \tilde{u} = u + \epsilon(u^{\mathrm{obs}} - G_\sigma * u^{\mathrm{obs}})$$

for some blur value σ and sufficiently small ϵ. By similar methods as above, we find

$$\begin{aligned}
\hat{\tilde{u}}(k) &= \hat{u}^{\mathrm{obs}}(k) + \epsilon \left(\mathrm{Id} - \mathrm{e}^{-\frac{\sigma^2 k^2}{2}} \right) \hat{u}^{\mathrm{obs}}(k) \\
&\simeq \left(\mathrm{Id} + \frac{\epsilon \sigma^2 k^2}{2} \right) \hat{u}^{\mathrm{obs}}(k) \\
\Rightarrow \tilde{u} &\simeq \left(\mathrm{Id} - \frac{\epsilon \sigma^2}{2} \Delta \right) u^{\mathrm{obs}}(x) \,.
\end{aligned}$$

We may choose to interpret the operators $\mathcal{E}_{\delta t}(\zeta)$ as a kind of "non-stationary unsharp masking".

For the presented problem of *non-stationary nonlinear* blind deconvolution/inverse filtering, we are not aware of any suitable classical methods. For a recent study that discusses backward diffusion, see [4].

2.3 Discretisation

We introduce the definition of a sparse matrix operator representing local non-stationary convolution

Definition 2 W is called a *Sparse Sub-Diagonal* (SSD) matrix if its nonzero entries are all on sub-diagonals corresponding to the local neighbourhood of pixels on its diagonal.

Furthermore, we are going to consider that a class of SSD matrices $\mathsf{W}(\zeta)$ with learned parameters ζ can be decomposed as $\mathsf{W}(\zeta) = \mathsf{S}(\zeta) + \mathsf{L}(\zeta)$ where S is *smoothing* and $\mathsf{L}(\zeta)$ is *zero-mean*, i.e. $\mathsf{L}(\zeta)$ has one zero eigenvalue such that its application to a constant image gives a zero valued image

$$\mathsf{L}(\zeta)\mathbb{1} = 0$$

In the following, we restrict ourselves to the typical 4-connected neighbourhood of pixels in dimension $d = 2$. For the numerical implementation, we have the Laplacian stencil

$$\Delta \quad \to \quad \mathsf{L}_\Delta = \begin{pmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{pmatrix}$$

from which we have that L_Δ is zero mean. Similarly, we will have for the numerical approximation of $\mathcal{E}_{\delta t}^{\mathrm{iso}}$ the matrix operator

$$\mathrm{Id} - \delta t \Delta \quad \to \quad \mathsf{E}_{\delta t}^{\mathrm{iso}} = \begin{pmatrix} & 0 & \\ 0 & 1 & 0 \\ & 0 & \end{pmatrix} - \begin{pmatrix} & \delta t & \\ \delta t & -4\delta t & \delta t \\ & \delta t & \end{pmatrix}.$$

Further we conjecture that in the numerical setting, $\mathcal{E}_{\delta t}(\zeta)$ is approximated by the sum of identity plus a SSD matrix operator as

$$\begin{aligned}
\mathcal{E}_{\delta t}(\zeta) \;&\sim\; \mathrm{Id} - \delta t \mathcal{L}(\zeta) \to \mathsf{Id} - \mathsf{L}_{\delta t}(\zeta) \\
&\sim\; \mathsf{E}_{\delta t}(\zeta) = \begin{pmatrix} & 0 & \\ 0 & 1 & 0 \\ & 0 & \end{pmatrix} - \delta t \begin{pmatrix} & \zeta_1 & \\ \zeta_2 & -\sum_i \zeta_i & \zeta_4 \\ & \zeta_3 & \end{pmatrix}.
\end{aligned} \tag{2.22}$$

In the presence of noise, the inverse operator $\mathcal{E}_{\delta t}(\zeta)$ can be explicitly regularised by addition of a smoothing operation

$$\tilde{\mathcal{E}}_{\delta t}(\zeta) = \mathcal{E}_{\delta t}(\zeta) + \alpha \mathcal{S} \to \mathsf{Id} - \mathsf{L}_{\delta t}(\zeta) + \alpha \mathsf{S} =: \mathsf{Id} - \mathsf{W}_{\delta t}(\zeta) \tag{2.23}$$

whereas in classical approaches to inverse filtering, the regularisation operator would be defined *a priori*, the approach in this paper is to learn the operator W and interpret it as the sum of a differentiating operator L and a (learned) regulariser S. This is discussed further in Sect. 4

3 Continuum Networks

Motivated by the previous section, we aim to build network architectures based on diffusion processes. We first discuss the notion of (neural) networks in a continuum setting for which we introduce the concept of a *continuum network* as a mapping between function spaces. That is, given a function on a bounded domain $\Omega \subset \mathbb{R}^d$ with $f \in L^p(\Omega)$, we are interested in finding a nonlinear parametrised operator $\mathcal{H}_\Theta : L^p(\Omega) \to L^p(\Omega)$ acting on the function f. We will consider in the following the case $p \in \{1, 2\}$; extensions to other spaces depend on the involved operations and will be the subject of future studies.

We will proceed by defining the essential building blocks of a continuum network and thence to discuss specific choices to obtain a continuum version of the most common convolutional neural networks. Based on this, we will then introduce our proposed architecture as a diffusion network in the next chapter.

3.1 Formulating a Continuum Network

The essential building blocks of a deep neural network are obviously the several layers of neurons, but since these have a specific notion in classical neural networks, see, for instance, [35], we will not use the term of neurons to avoid confusion. We rather introduce the concept of layers and channels as the building blocks of a *continuum network*. In this construction, each layer consists of a set of functions on a product space and each function represents a channel.

Definition 3 (*Layer and channels*) For $k \in \mathbb{N}_0$, let $F_k = \{f_1^k, f_2^k, \cdots, f_I^k\}$ be a set of functions $f_i^k \in L^p(\Omega)$ for $i \in \mathbb{I} = \{1, \ldots, I\}$, $I \geq 1$. Then, we call: F_k the layer k with I channels and corresponding index set $\mathbb{I}$.

The continuum network is then built by defining a relation or operation between layers. In the most general sense, we define the concept of a layer operator for this task.

Definition 4 (*Layer operator*) Given two layers F_k and F_t, $k \neq t$, with channel index set $\mathbb{I}$, $\mathbb{J}$, respectively, we call the mapping $\mathcal{H} : \bigotimes_{\mathbb{I}} L^p(\Omega) \to \bigotimes_{\mathbb{J}} L^p(\Omega)$ with

$$\mathcal{H}F_k = F_t$$

a layer operator. If the layer operator depends on a set of parameters Θ, then we write $\mathcal{H}_\Theta$.

We note that for simplicity, we will not index the set of parameters, i.e. Θ generally stands for both all involved parameters of each layer separately, or the whole network. The classical structure for layer operators follows the principle of affine linear transformations followed by a nonlinear operation. Ideally, the affine linear transformation should be parameterisable by a few parameters, whereas the nonlinear operation is often fixed and acts pointwise. A popular choice is the maximum operator also called the "Rectified Linear Unit":

$$\mathrm{ReLU} : L^p(\Omega) \to L^p(\Omega), \quad f \mapsto \max(f, 0).$$

The continuum network is then given by the composition of all involved layer functions. For example, in monochromatic imaging applications, we typically have an input image f_0 and a desired output f_K with several layer functions in-between that perform a specific task such as denoising or sharpening. In this case, the input and output consist of one channel, i.e. $|F_0| = |F_K| = 1$; consequently, for colour images (in RGB), we have $|F_0| = |F_K| = 3$.

3.2 Continuum Convolutional Networks

Let us now proceed to discuss a specific choice for the layer operator, namely convolutions. With this choice, we will obtain a continuum version of the widely used convolutional neural networks, which we will call here a *continuum convolutional network*, to avoid confusion with the established convolutional neural networks (CNN). We note that similar ideas have been addressed as well in [2].

Let us further consider linearly ordered network architectures that means each layer operator maps between consecutive layers. The essential layer operator for a continuum convolutional network is then given by the following definition.

Definition 5 (*Convolutional layer operator*) Given two layers F_{k-1} and F_k with channel index set $\mathbb{I}$, $\mathbb{J}$, respectively, let $b_j \in \mathbb{R}$ and $\omega_{i,j} \in L^p(\Omega)$, with compact support in Ω, be the layer operator's parameters for all $i \in \mathbb{I}$, $j \in \mathbb{J}$. We call $\mathcal{C}_{\Theta,\varphi}^{(k)}$ the convolutional layer operator for layer k, if for each output channel

$$\mathcal{C}_{\Theta,\varphi}^{(k)} F_{k-1} = \varphi\left[b_j + \sum_{i \in \mathcal{I}} \omega_{i,j} * f_i^{k-1}\right] = f_j^k, \ j \in \mathbb{J}, \tag{3.1}$$

with a pointwise nonlinear operator $\varphi : L^p(\Omega) \to L^p(\Omega)$.

If the layer operator does not include a nonlinearity, we write $\mathcal{C}_{\Theta,\mathrm{Id}}$. Now, we can introduce the simplest convolutional network architecture by applying $K \geq 1$ convolutional layer operators consecutively.

Definition 6 (K*-layer Continuum Convolutional Network*) Let $K \geq 1$, then we call the composition of K convolutional layer operator, denoted by $\mathcal{C}_\Theta^K$, a K-layer continuum convolutional network, such that

$$\mathcal{C}_{\Theta,\varphi}^K = \mathcal{C}_{\Theta,\varphi}^{(K)} \circ \cdots \circ \mathcal{C}_{\Theta,\varphi}^{(1)}, \quad \mathcal{C}_\Theta^K F_0 = F_K.\text{`} \tag{3.2}$$

In the following, we will also refer to a K-layer CNN as the practical implementation of a K-layer continuum convolutional network. A popular network architecture that extends this simple idea is given by a residual network (ResNet) [18], that is, based on the repetition of a 2-layer CNN with a residual connection, that consists of addition. That is, the network learns a series of additive updates to the input. The underlying structure in ResNet is the repeated application of the following *residual block* given by

$$\mathcal{R}_{\Theta,\varphi} = \mathcal{C}_{\Theta,\mathrm{Id}}^{(2)} \circ \mathcal{C}_{\Theta,\varphi}^{(1)} + \mathrm{Id}, \quad \mathcal{R}_{\Theta,\varphi} F_0 = F_2. \tag{3.3}$$

Note that the second convolutional layer does not include a nonlinearity. Furthermore, it is necessary that $|F_0| = |F_2|$, but typically, it is often chosen such that $|F_0| = |F_1| = |F_2|$. The full continuum ResNet architecture can then be summarised as follows. Let $K \geq 1$, then the composition

of K residual blocks, denoted by $\mathcal{R}^K_{\Theta,\varphi}$, defines a K-block continuum ResNet

$$\mathcal{R}^K_{\Theta,\varphi} = \mathcal{C}^{(K+1)}_{\Theta,\varphi} \circ \mathcal{R}^{(K)}_{\Theta,\varphi} \circ \cdots \circ \mathcal{R}^{(1)}_{\Theta,\varphi} \circ \mathcal{C}^{(0)}_{\Theta,\varphi}. \tag{3.4}$$

Note that the two additional convolutional layers in the beginning and end are necessary to raise the cardinality of the input/output layer to the cardinality needed in the residual blocks. A complete K-block ResNet then consists of $2(K+1)$ layers. Note that in the original work [18], the network was primarily designed for an image classification task rather than image-to-image mapping that we consider here.

4 DiffNet: Discretisation and Implementation

In this section, we want to establish a layer operator based on the diffusion processes discussed in chapter 2. This means that we now interpret the layers F_k of the continuum network as time states of the function $u : \Omega \times \mathbb{R}_+ \to \mathbb{R}$, where u is a solution of the diffusion Eq. (2.1). In the following, we assume single-channel networks, i.e. $|F_k| = 1$ for all layers. Then, we can associate each layer with the solution u such that $F_k = u^{(k)} = u(x, t = t_k)$. To build a network architecture based on the continuum setting, we introduce the layer operator versions of (2.10), and (2.20):

Definition 7 (*Diffusion and filtering layer operator*) Given two layers F_k and F_{k-1}, such that $F_k = u(x, t_k)$ and $F_{k-1} = u(x, t_{k-1})$, then a diffusion layer operator $\mathcal{D}_\Theta$, with parameters $\Theta = \{\gamma, \delta t\}$, is given by

$$\begin{aligned} \mathcal{D}_\Theta F_{k-1} &= \mathcal{D}^{\text{Expl}}_{\delta t}(\gamma^{(k-1)})u^{(k-1)} \\ &= (\text{Id} + \delta t \mathcal{L}(\gamma^{(k)}))u^{(k-1)} = F_k. \end{aligned} \tag{4.1}$$

Similarly, an inverse filtering layer operator with parameters $\Theta = \{\zeta, \delta t\}$ is given by

$$\begin{aligned} \mathcal{E}_\Theta F_{k-1} &= \mathcal{E}_{\delta t}(\zeta^{(k-1)})u^{(k-1)} \\ &= u^{(k-1)} - \delta t \int_{\mathbb{R}^d} \zeta(x, y)u^{(k-1)}(y)\mathrm{d}y = F_k. \end{aligned} \tag{4.2}$$

Note that this formulation includes a learnable time step and hence the time instances that each layer represents changes. That also means that a stable step size is implicitly learned, if there are enough layers. In the following, we discuss a few options on the implementation of the above layer operator, depending on the type of diffusivity.

Remark 4 The assumption of a single-channel network, i.e. $|F_k| = 1$ for all k, can be relaxed easily, either by assuming $|F_k| = m$ for some $m \in \mathbb{N}$ and all layers, or by introducing a channel mixing as in the convolutional operator (3.1).

As a natural application, we could consider RGB or hyperspectral images as a multichannel input. In that case, the filters would become a tensor representing both intra- and inter-channel mixing but still modelled as a diffusion process—see, for example, [9].

4.1 Discretisation of a Continuum Network

Let us briefly discuss some aspects on the discretisation of a continuum network; we first start with affine linear networks, such as the convolutional networks discussed in Sect. 3.2. Rather than discussing the computational implementation of a CNN, (see, for example, the comprehensive description in [10]), we concentrate instead on an algebraic matrix–vector formulation that serves our purposes.

For simplicity, we concentrate on the two-dimensional $d = 2$ case here. Let us then assume that the functions f_i in each layer are represented as a square n-by-n image and we denote the vectorised form as $\mathbf{f} \in \mathbb{R}^{n^2}$. Then, any linear operation on $\mathbf{f}$ can be represented by some matrix A; in particular, we can represent convolutions as a matrix.

We now proceed by rewriting the convolutional operation (3.1) in matrix–vector notation. Given two layers F_k and F_{k-1} with channel index set $\mathbb{I}$, $\mathbb{J}$, respectively, then we can represent the input layer as vector $\mathbf{F}_{k-1} \in \mathbb{R}^{Jn^2}$ and similarly the output layer $\mathbf{F}_k \in \mathbb{R}^{In^2}$. Let $\mathsf{A}_i \in \mathbb{R}^{n^2 \times Jn^2}$ represent the sum of convolutions in (3.1), then we can write the layer operator in the discrete setting as matrix–vector operation by

$$\mathbf{f}^k_i = \varphi(b_i \mathbb{1} + \mathsf{A}_i \mathbf{F}_{k-1}),$$

where $\mathbb{1}$ denotes a vector of ones with suitable size. Furthermore, following the above notation, we can introduce a stacked matrix $\mathsf{A} \in \mathbb{R}^{In^2 \times Jn^2}$ consisting of all A_i and a vector $\mathsf{b} \in \mathbb{R}^I$ consisting of all biases. Then, we can represent the whole convolutional layer in the discrete setting as

$$\mathbf{F}_k = \varphi(b \otimes \mathbb{1} + \mathsf{A}\mathbf{F}_{k-1}). \tag{4.3}$$

Now, the parameters of each layer are contained in the matrix A and vector b.

4.2 Learned Forward and Inverse Operators

Let us now discuss a similar construction for the diffusion layers. For the implementation of the diffusion network, we consider the explicit formulation in (2.10) with the differential operator $\mathcal{L}(\gamma^{(k)}) = \nabla \cdot \gamma^{(k)} \nabla$ approximated by the stencil (including the time step δt)

$$\mathsf{L}_{\delta t}(\gamma^{(k)}) = \delta t \begin{pmatrix} & \gamma_1^{(k)} & \\ \gamma_2^{(k)} & -\sum_i \gamma_i^{(k)} & \gamma_4^{(k)} \\ & \gamma_3^{(k)} & \end{pmatrix}. \tag{4.4}$$

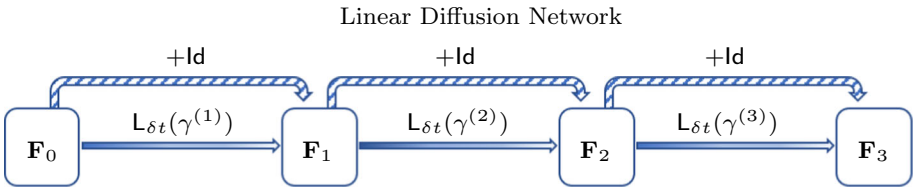

Fig. 1 Illustration for a linear three-layer diffusion network. In this case, we learn the filters γ as the diffusivity for each layer explicitly

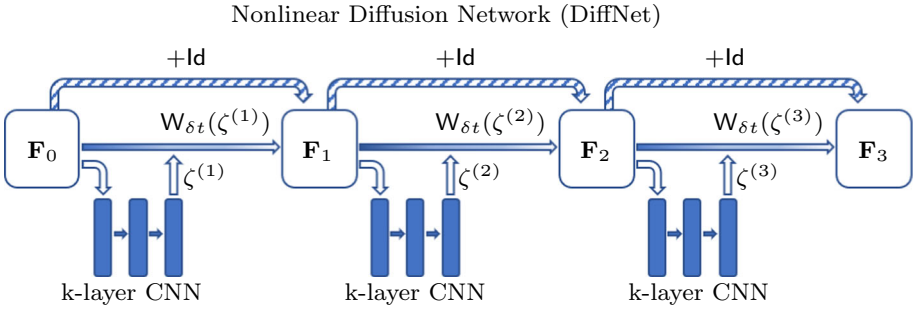

Fig. 2 Illustration for a nonlinear three-layer diffusion network. Here, the filters ζ are implicitly estimated by a small k-layer CNN and then applied to the image in the filtering layer

We use zero Neumann boundary conditions on the domain boundary

$$\partial_v u \;=\; 0 \text{ on } \partial\Omega \times (0, T]. \tag{4.5}$$

Then, we can represent (4.1) by

$$\mathbf{F}_k = (\mathsf{Id} + \mathsf{L}_{\delta t}(\gamma))\mathbf{F}_{k-1}. \tag{4.6}$$

The basis of learning a diffusion network is now given as estimating the diagonals of $\mathsf{L}_{\delta t}(\gamma)$ and the time step δt. This can be done either explicitly as for the CNN or indirectly by an estimator network, as we will discuss next.

4.3 Formulating DiffNet

Let us first note that if we construct our network by strictly following the update in (4.6), we restrict ourselves to the forward diffusion. To generalise the inverse problem, we consider

$$\mathsf{W}_{\delta t}(\zeta) = \delta t \begin{pmatrix} & \zeta_1 & \\ \zeta_2 & -\zeta_5 & \zeta_4 \\ & \zeta_3 & \end{pmatrix}. \tag{4.7}$$

Additionally, there are two fundamentally different cases for the diffusivity γ we need to consider before formulating a network architecture to capture the underlying behaviour. These two cases are

(i) Linear diffusion; spatially varying and possible time dependence, $\gamma = \gamma(x, t)$.
(ii)) Nonlinear diffusion; diffusivity depending on the solution u, $\gamma = \gamma(u(x, t))$.

In the first case, we could simply try to learn the diffusivity explicitly, to reproduce the diffusion process. In the second case, this is not possible, and hence, an estimation of the diffusivity needs to be performed separately in each time step from the image itself, before the diffusion step can be performed. This leads to two conceptually different network architectures.

The linear case (i) corresponds to the diffusion layer operator (4.1) and is aimed to reproduce a linear diffusion process with fixed diffusivity. Thus, learning the mean-free filter suffices to capture the physics. The resulting network architecture is outlined in Fig. 1. Here, the learned filters can be directly interpreted as the diffusivity of layer k and are then applied to $\mathbf{F}_{k-1}$ to produce $\mathbf{F}_k$.

In the nonlinear case (ii), we follow the same update structure, but now the filters are not learned explicitly; they are rather estimated from the input layer itself, as illustrated in Fig. 2. Furthermore, since this architecture is designed for inversion of the nonlinear diffusion process, we employ the generalised stencil $\mathsf{W}_{\delta t}(\zeta)$. Then, given layer $\mathbf{F}_k$, the filters ζ are estimated by a small CNN from $\mathbf{F}_{k-1}$, which are then applied following an explicit update as in (4.6) to produce $\mathbf{F}_k$. Note that the diagonals in the update matrix are produced by the estimation CNN. We will refer to this nonlinear filtering architecture as the *DiffNet* under consideration in the rest of this study.

In contrast to classical CNN architectures, the proposed DiffNet is nonlinear by design, and hence, no additional nonlinearities are necessary. Compared to previous approaches, such as [6], we note that the parameters of the estimating CNN are not used to process the image directly, but rather to produce the filters ζ for the update matrix only.

Comparing 'Diffnet' to 'PDE-NET2.0' [27], in the latter, the training assumes that full time-series data $u(x, t)$ are available, and the PDE is approximated by a forward

Euler method with appropriate stability constraints. In our approach, only initial and final conditions are assumed to be known; the coefficients of the PDE are spatially varying, and both a *forward* and an *inverse* problem can be learned, and the latter requires regularisation which is learned simultaneously with the PDE coefficients.

4.3.1 Implementation

The essential part for the implementation of a diffusion network is to perform the update (4.6) with either $\mathsf{L}_{\delta t}(\gamma)$ or $\mathsf{W}_{\delta t}(\zeta)$. For computational reasons, it is not practical to build the sparse diagonal matrix and evaluate (4.6); we rather represent the filters γ and ζ as an $n \times n$-image and apply the filters as pointwise matrix–matrix multiplication to a shifted and cropped image, according to the position in the stencil. This way, the zero Neumann boundary condition (4.5) is also automatically incorporated.

For the linear diffusion network, we would need to learn the parameter set Θ, consisting of filters and time steps, explicitly. This has the advantage of learning a global operation on the image where all parameters are interpretable, but it comes with a few disadvantages. First of all, in this form, we are limited to linear diffusion processes and a fixed image size. Furthermore, the parameters grow with the image size, i.e. for an image of size $n \times n$, we need $5n^2$ parameters per layer. Thus, applications may be limited.

For the nonlinear architecture of DiffNet, where the filters depend on the image at each time step, we introduced an additional estimator consisting of a K-layer CNN. This CNN gets an image, given as layer $\mathbf{F}_k$, as input and estimates the filters ζ. The architecture for this K-layer CNN as estimator is chosen to be rather simplistic, as illustrated in Fig. 3. The input $\mathbf{F}_k$ consists of one channel, which is processed by $K-1$ convolutional layers with 32 channels and a ReLU nonlinearity, followed by the last layer without nonlinearity and five channels for each filter, which are represented as matrices of the same size as the input $\mathbf{F}_k$. In particular, for the chosen filter size of 3×3, we have exactly $9 \cdot (32 + 32 \cdot 5 + 32^2 \cdot (K-2))$ convolutional parameters and $32 \cdot (K-1) + 5$ biases per diffusion layer. That is for a five-layer CNN, we have 29.509 parameters independent of image size.

5 Computational Experiments

In the following, we will examine the reconstruction capabilities of the proposed DiffNet. The experiments are divided into a simple case of deconvolution, where we can examine the learned features and a more challenging problem of recovering an image from its nonlinear diffused and noise-corrupted version.

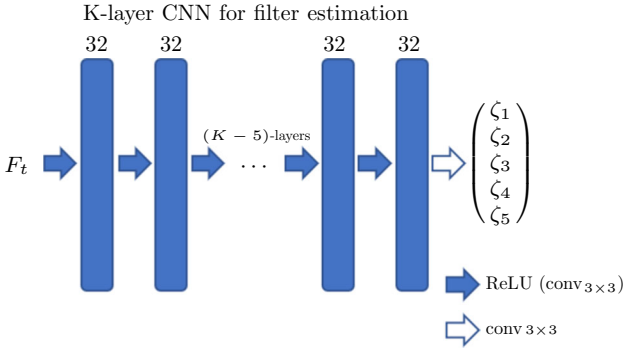

Fig. 3 Architecture of the K-layer CNN used as diffusivity estimator in the nonlinear diffusion network (DiffNet)

5.1 Deconvolution with DiffNet

We first examine a simple deconvolution experiment to determine what features the DiffNet learns in an inverse problem. For this task, we will only consider deconvolution without noise.

The forward problem is given by (2.1) with zero Neumann boundary condition (4.5) and constant diffusivity $\gamma \equiv 1$. For the experiment, we choose $T = 1$, which results in a small uniform blurring, as shown in Fig. 4. We remind that for the isotropic diffusion, the forward model is equivalent to convolution in space with the kernel $G_{\sqrt{2T}}$, see (2.3). As it is also illustrated in Fig. 4, convolution in the spatial domain is equivalent to multiplication in Fourier domain. In particular, high frequencies get damped and the convolved image is dominated by low frequencies. Hence, for the reconstruction task without noise, we essentially need to recover the high frequencies.

The training and test data for DiffNet consist of simple discs of varying radius and contrast. The training set consists of 1024 samples and the test set of an additional 128, each of size 64×64. The network architecture is chosen following the schematic in Fig. 2, with three diffusion layers and a final projection to the positive numbers by a ReLU layer. The filter estimator is given by a four-layer CNN, as described in Sect. 4.3.1. All networks were implemented in Python with TensorFlow [1].

The input to the network is given by the convolved image without noise, and we have minimised the ℓ^2-loss of the output to the ground-truth image. The optimisation is performed for about 1000 epochs in batches of 16 with the Adam algorithm and initial learning rate of 4×10^{-4} and a gradual decrease to 10^{-6}. Training on a single Nvidia Titan Xp GPU takes about 24 min. The final training and test error are both at a PSNR of 86.24, which corresponds to a relative ℓ^2-error of 2.5×10^{-4}. We remind that this experiment was performed without noise.

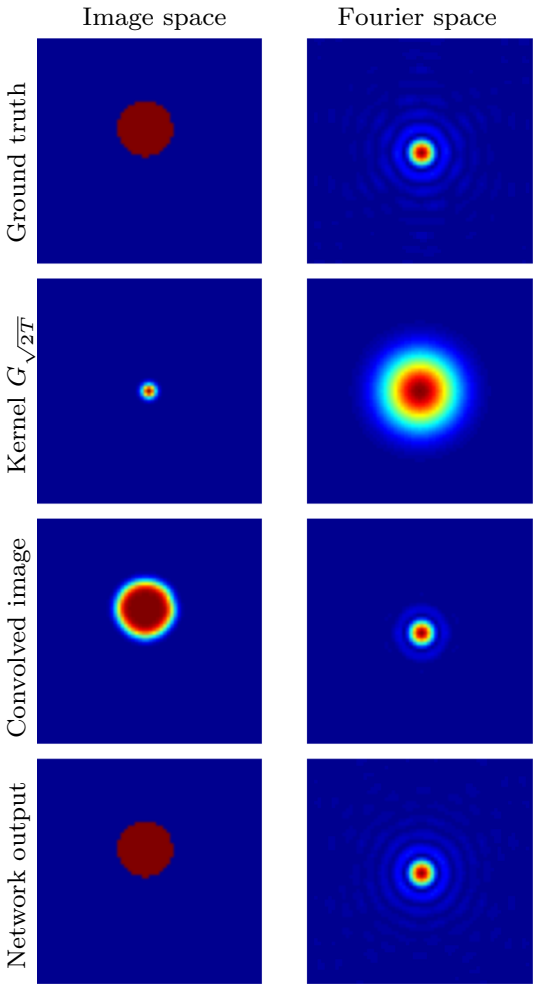

Fig. 4 Illustration of the deconvolution problem for a simple ball. Left column shows the image space, and the right column shows the corresponding absolute value of the Fourier coefficients. All images are plotted on their own scale

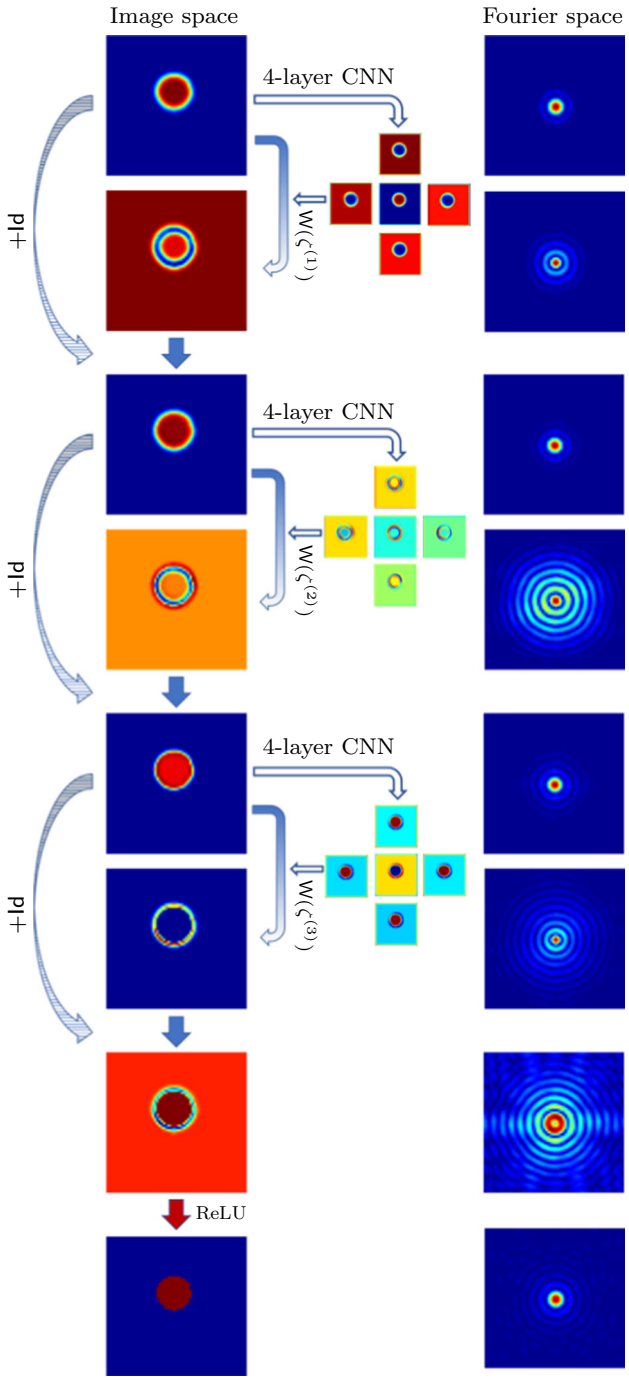

Fig. 5 Illustration of the deconvolution process with three layers of DiffNet. The left column shows the processed image and intermediate steps. The right column shows the corresponding absolute value of Fourier coefficients. All images are plotted on their own scale

The result of the network and intermediate updates for one example from the test data are illustrated in Fig. 5. We also show the filters $\zeta^{(k)}$ computed as the output of the trained CNN in each layer, $k = 1, 2, 3$. The output of the last diffusion layer $\mathbf{F}_3$ is additionally processed by a ReLU layer to enforce positivity in the final result. It can be seen that the network gradually reintroduces the high frequencies in the Fourier domain; especially, the last layer mainly reintroduces the high frequencies to the reconstruction. It is interesting to see that the learned filters follow indeed the convention that the central filter is of different signs than the directional filters. This enforces the assumption that the filter consists of a mean-free part and a regularising part, which should be small in this case, since we do not have any noise in the data. Lastly, we note that the final layer, before projection to the positive numbers, has a clearly negative part around the target, which will be cut off resulting in a sharp reconstruction of the ball.

5.2 Nonlinear Diffusion

Let us now consider the nonlinear diffusion process with the Perona–Malik filter function [30] for (2.1) with zero Neumann boundary condition (4.5). In this model, the diffusivity is given as a function of the gradient

Fig. 6 Samples from the test data for learning the inversion of nonlinear diffusion without noise. Mean PSNR for reconstructed test data with DiffNet is: 63.72

$$\gamma(|\nabla u|^2) = \frac{1}{1 + |\nabla u|^2/\lambda^2} \tag{5.1}$$

with contrast parameter $\lambda > 0$. We mainly concentrate here on the inverse problem of restoring an image that has been diffused with the Perona–Malik filter and contaminated by noise.

For the experiments, we have used the test data from the STL-10 database [7], which consists of 100,000 RGB images with resolution 96×96. These images have been converted to grey scale and divided to 90,000 for training and 10,000 for testing. The obtained images were then diffused for four time steps with $\delta t = 0.1$ and $\lambda = 0.2$. A few sample images from the test data with the result of the diffusion are displayed in Fig. 6. The task is then to revert the diffusion process with additional regularisation to deal with noise in the data.

For all experiments, we have used the same network architecture of DiffNet using the architecture as illustrated in Fig. 2. By performing initial tests on the inversion without noise, we have found that five diffusion layers with a four-layer CNN, following the architecture in 3, gave the best trade-off between reconstruction quality and network size. Increasing the amount of either layers led to minimal increase in performance. Additionally, we have used a ReLU layer at the end to enforce nonnegativity of the output, similarly to the last experiment. We emphasise that this architecture was used for all experiments and hence some improvements for the high-noise cases might be expected with more layers. All networks were trained for 18 epochs, with a batch size of 16, and ℓ^2-loss. For the optimisation, we have used the Adam algorithm with initial learning rate of 2×10^{-3} and a gradual decrease to 4×10^{-6}. Training on a single Nvidia Titan Xp GPU takes about 75 min.

As benchmark, we have performed the same experiments with a widely used network architecture known as U-Net [32]. This architecture has been widely applied in inverse problems [3,12,21,22], even for applications where it is theoretically unsuitable, and hence can be considered as an established benchmark. It is mainly used to post-process initial directly reconstructed images from undersampled or noisy data, for instance, by filtered back-projection in X-ray tomography or the inverse Fourier transform in magnetic resonance imaging [16]. The network architecture we are using follows the publication [21] and differs from the original mainly by a residual connection at the end. That means the network is trained to remove noise and undersampling artefacts from the initial reconstruction. In our context, the network needs to learn how to remove noise and reintroduce edges. For training, we have followed a similar protocol as for DiffNet. The only difference is that we started with an initial learning rate of 5×10^{-4} with a gradual decrease to 2×10^{-5}. Training of U-Net takes about 3 h.

The reconstruction results, for some samples of the test data, with DiffNet can be seen in Fig. 6 for the case without noise and in Fig. 7 for 1% noise on the diffused images. A comparison of the reconstruction results with U-Net and DiffNet is shown in Fig. 8 for the test without noise and in Fig. 9 for 1% noise. Qualitatively, the reconstructed images are very similar, as can be seen in the residual images in the last column. The leftover noise pattern for both networks is

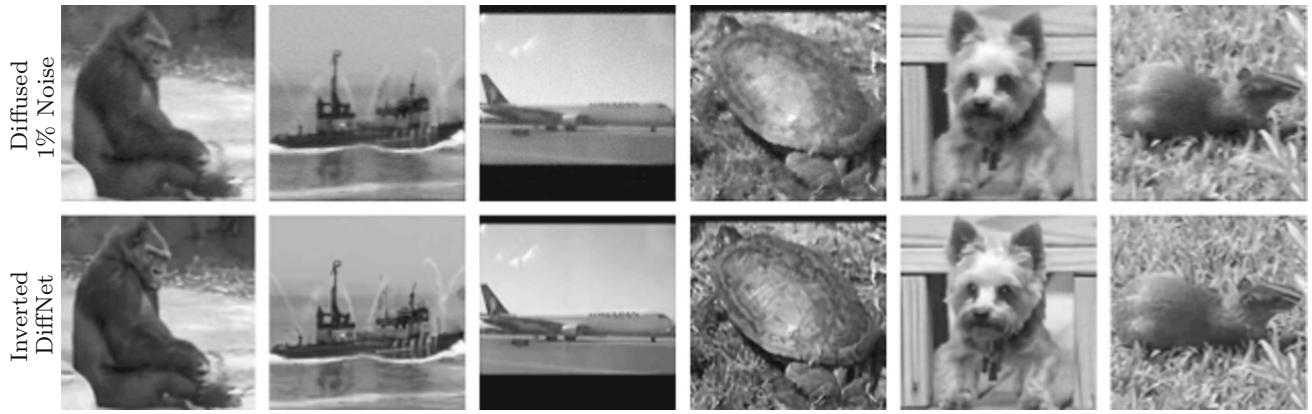

Fig. 7 Samples from the test data for learning the inversion of nonlinear diffusion with 1% noise. Mean PSNR for reconstructed test data with DiffNet is: 34.21

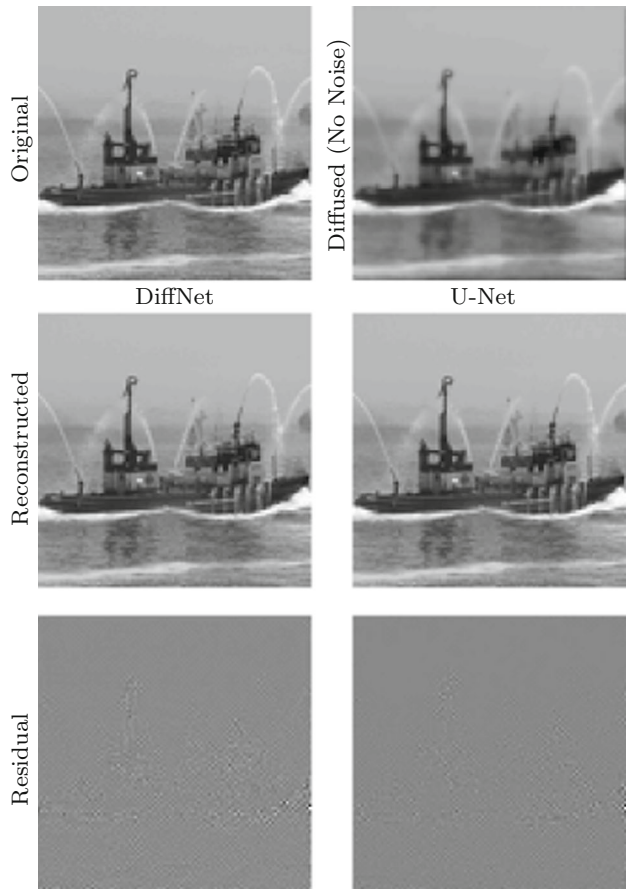

Fig. 8 Comparison of reconstruction quality for reconstruction from nonlinear diffused image without noise. Both networks are trained on the full set of 90,000 images. PNSR: DiffNet 65.34, U-Net 61.08

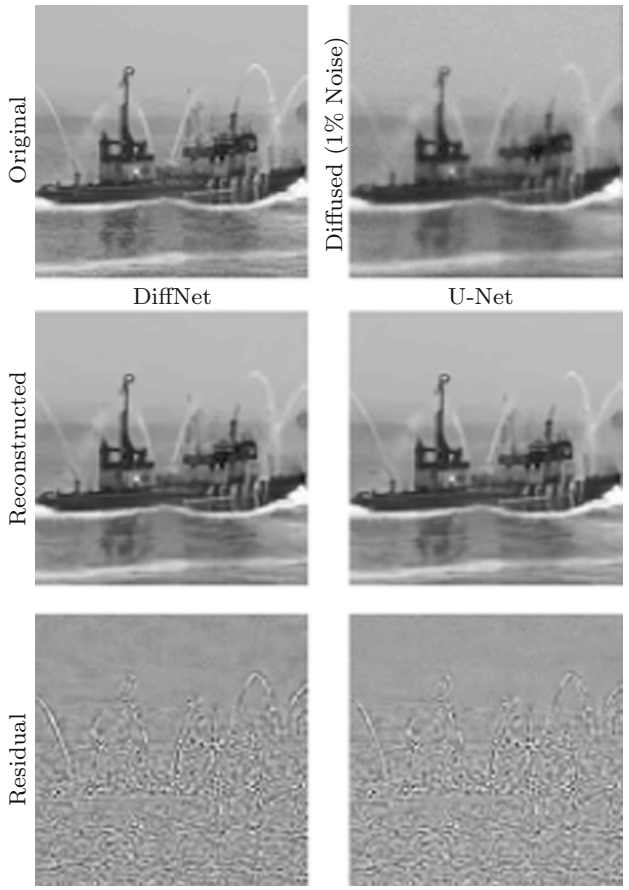

Fig. 9 Comparison of reconstruction quality for reconstruction from 1% noise contaminated nonlinear diffused image. Both networks are trained on the full set of 90,000 images. PNSR: DiffNet 34.96, U-Net 35.27

concentrated on the fine structures of the ship. Quantitatively, for the noise-free experiment, DiffNet has an increase of 4 dB in PSNR compared to the result of U-Net, 65.34 (DiffNet) compared to 61.08 (U-Net). For the case with 1% noise, the quantitative measures are very similar. Here, U-Net has a slightly higher PSNR with 35.27 compared to DiffNet with 34.96. A thorough study of reconstruction quality of both networks follows in the next section as well as some interpretation of the learned features in DiffNet.

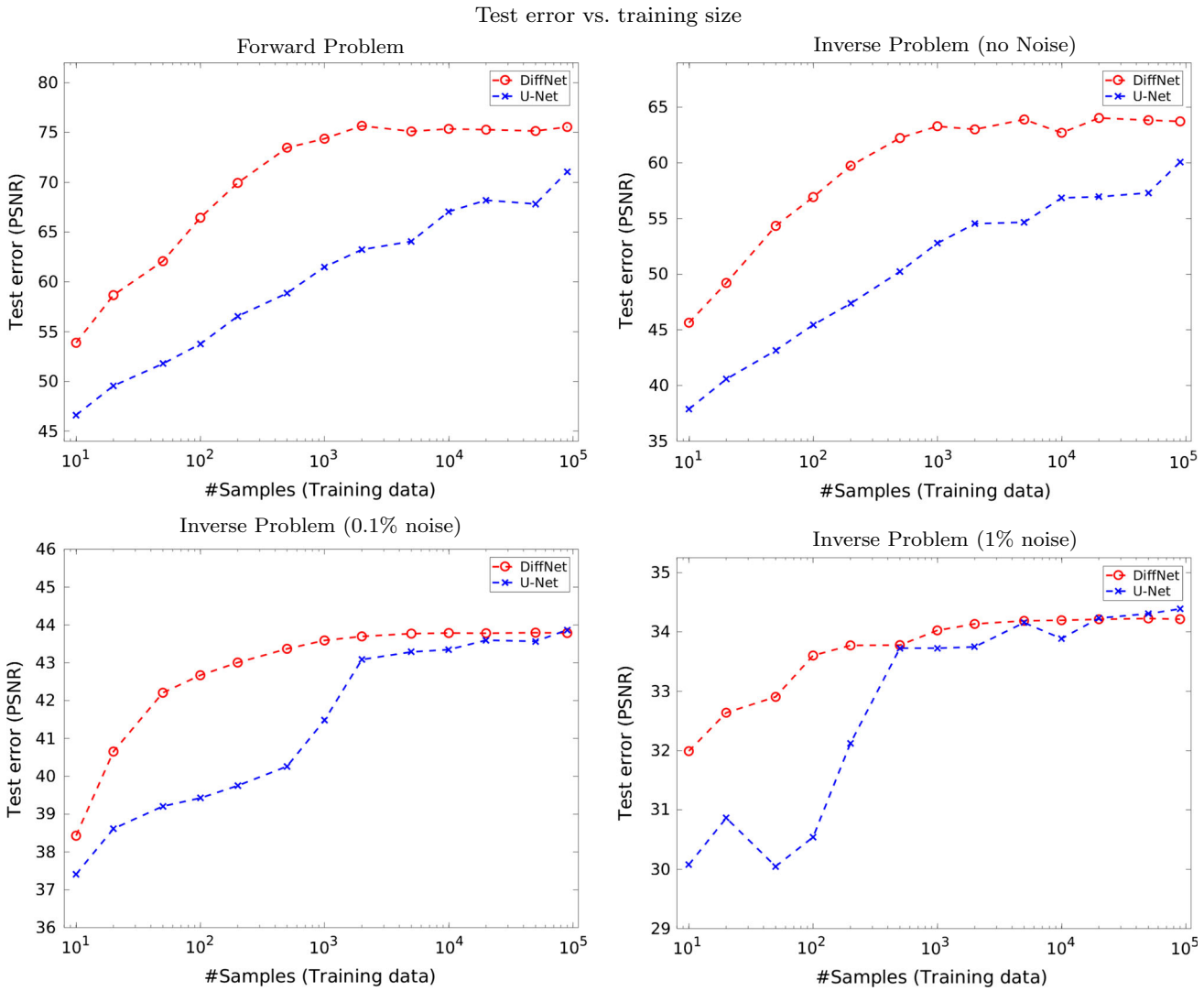

Fig. 10 Generalisation plot for the forward and inverse problems of nonlinear diffusion and varying noise levels. Test error depending on the amount of training data, for both DiffNet and U-Net

6 Discussion

First of all, we note that the updates in DiffNet are performed explicitly and that the CNN in the architecture is only used to produce the filters ζ. This means that DiffNet needs to learn a problem-specific processing, in contrast to a purely data-driven processing in a CNN. Consequently, the amount of necessary learnable parameters is much lower. For instance, the five-layer DiffNet used for inversion of the nonlinear diffusion in Sect. 5.2 has 101,310 parameters, whereas the used U-Net with a filter size of 3×3 has a total of 34,512,705 parameters, i.e. DiffNet uses only $\sim$ 0.3% of parameters compared to U-Net, and hence, the learned features can be seen to be much more explicit. In the following, we discuss some aspects that arise from this observation, such as generalisability and interpretability.

6.1 Generalisability

To test the generalisation properties of the proposed DiffNet, we have performed similar experiments as shown in Sect. 5.2 for nonlinear diffusion, but with increasing amounts of training data. Under the assumption that DiffNet learns a more explicit update than a classic CNN, we would expect also to require less training data to achieve a good test error. To certify this assumption, we have examined four settings of nonlinear diffusion with the Perona–Malik filter: learning the forward model, learning to reconstruct from the diffused image without noise, as well as with 0.1% and 1% noise. We then created training datasets of increasing size from just 10 samples up to the full size of 90,000. For all scenarios, we have trained DiffNet and U-Net following the training protocol described in 5.2. Additionally, we have aborted the

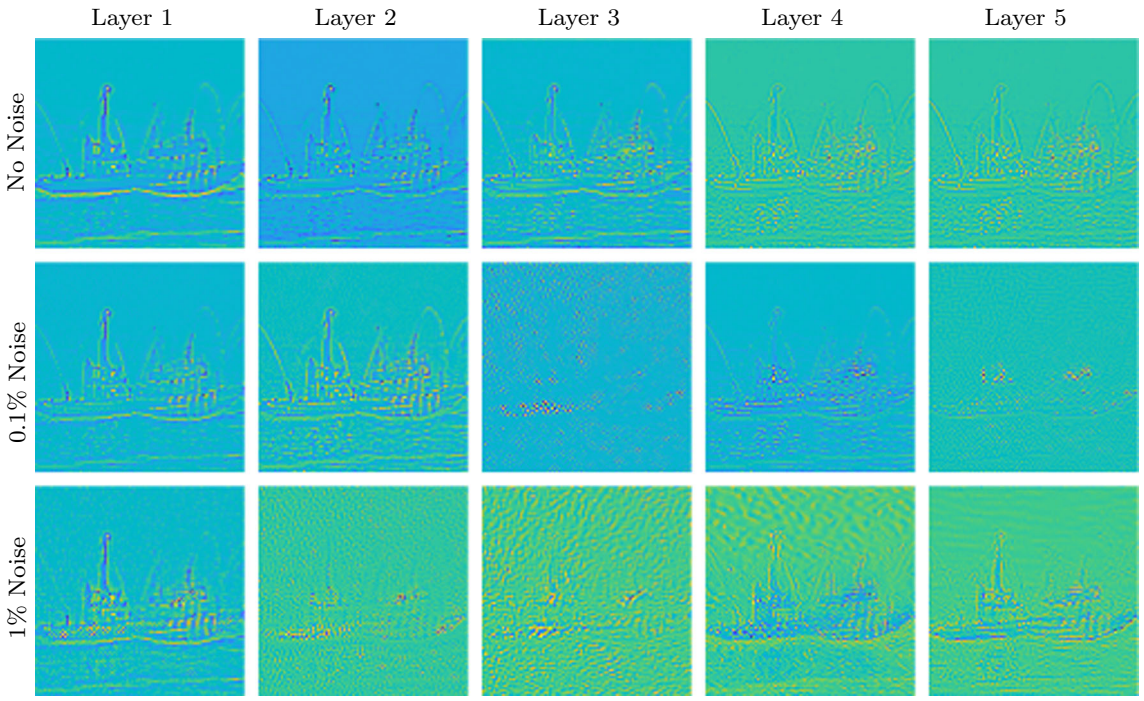

Fig. 11 Filter updates $\sum_{i=1}^{4} \zeta_i - \zeta_5$ for different noise levels. Each image is displayed on its own scale

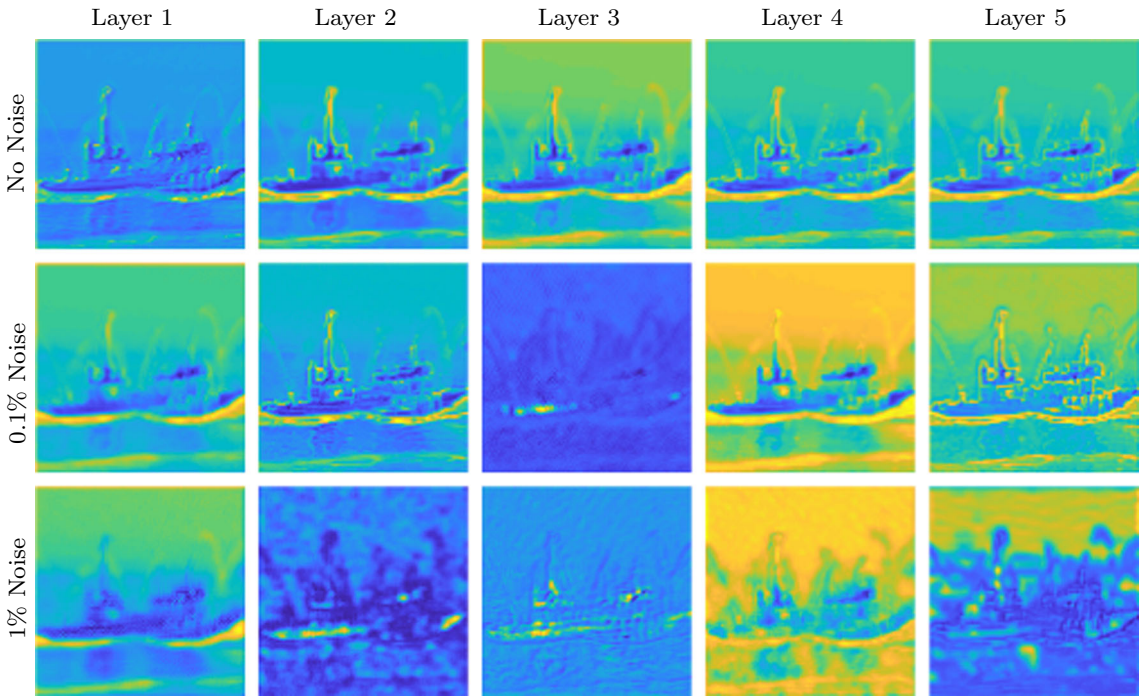

Fig. 12 Obtained diagonal filters ζ_5 for different noise levels. Each filter is displayed on its own scale

procedure when the networks started to clearly overfit the training data.

Results for the four scenarios are shown in Fig. 10. Most notably DiffNet outperforms U-Net clearly for the forward problem and the noise-free inversion, by 4dB and 3dB, respectively. For the noisy cases, both networks perform very similar for the full training data size of 90,000. The biggest difference overall is that DiffNet achieves its maximum test error already with 500–1000 samples independent of the noise case, whereas the U-Net test error saturates ear-

lier with higher noise. In conclusion, we can say that for the noisy cases, both networks are very comparable in reconstruction quality, but for small amounts of data, the explicit nature of DiffNet is clearly superior.

6.2 Interpretation of Learned Filters

Since all updates are performed explicitly with the output from the filter estimation CNN, we can interpret some of the learned features. For this purpose, we show the filters for the ship image from Sect. 5.2 for the three inversion scenarios under consideration. In Fig. 11, we show the sum of all learned filters, i.e. $\sum_{i=1}^{4} \zeta_i - \zeta_5$. If the network would only learn the mean-free differentiating part, then these images should be zero. This implies that the illustrated filters in Fig. 11 can be related to the learned regularisation $\mathsf{S}(\zeta)$. Additionally, we also show the diagonal filters ζ_5 in Fig. 12.

We would expect that with increasing noise level, the filters will incorporate more smoothing to deal with the noise; this implies that the edges get wider with increasing noise level. This can be nicely observed for the diagonal filters in Fig. 12. For the smoothing in Fig. 11, we see that the first layer consists of broader details and edges that are refined in the noise-free case for increasing layers. In the noisy case, the latter layers include some smooth features that might depict the regularisation necessary in the inversion procedure. It is generally interesting to observe that the final layer shows very fine local details, necessary to restore fine details for the final output.

Finally, we have computed training data of a wider noise range to examine the regularisation properties of the learned network. For this, we have taken the full 90,000 training samples and contaminated the diffused image with noise ranging from 0.01 to 10% noise. As we conjectured in Sect. 2.3, the learned update filters can be decomposed to a mean-free operation and a smoothing part $\mathsf{W}(\zeta) = \mathsf{L}(\zeta) + \mathsf{S}(\zeta)$. This implies that the magnitude of $\mathsf{S}(\zeta)$ has to increase with higher noise. To examine this conjecture, we have taken (fixed) 32 samples from the reconstructed test data for each noise level and computed an estimate of S as the sum $\sum_{i=1}^{4} \zeta_i - \zeta_5$, i.e. the deviation from the mean-free part. Furthermore, we use the mean of the absolute value of S over the whole image as an estimator of α. The resulting graph of smoothing versus noise level is shown in Fig. 13. As we have conjectured, the estimate of α increases clearly with the noise level, and hence, we believe our interpretation of the learned filters as the composition of a mean-free part and a smoothing necessary for ill-posed inverse problems is valid.

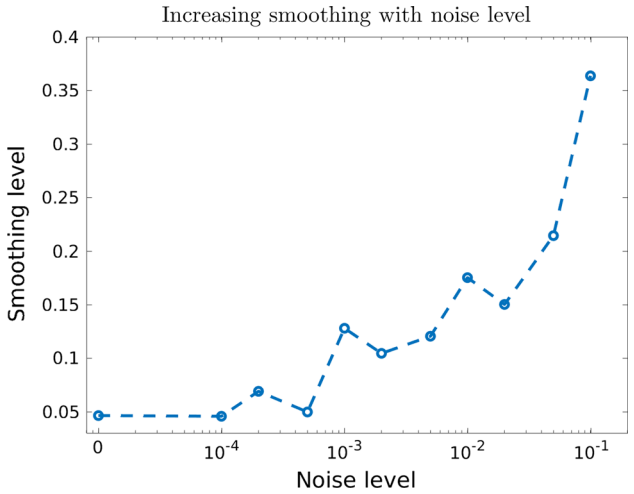

Fig. 13 Estimate of the smoothing level α for increasing noise in the inverse problem. Computed over a sample of 32 images from the test data

7 Conclusions

In this paper, we have explored the possibility to establish novel network architectures based on physical models other than convolutions; in particular, we concentrated here on diffusion processes. As main contributions, we have introduced some nonlinear forward mappings, modelled through learning rather than just through PDEs or integral transforms. We have reviewed (regularised) inverse diffusion processes for inverting such maps. In particular, we have conjectured that these inverse diffusion processes can be represented by local non-stationary filters, which can be learned in a network architecture. More specific, these local filters can be represented by a sparse sub-diagonal (SSD) matrix and hence efficiently used in the discrete setting of a neural network. We emphasise that even though we have concentrated this study on a specific structure for these SSD matrices based on diffusion, other (higher order) models can be considered.

We obtain higher interpretability of the network architecture, since the image processing is explicitly performed by the application of the SSD matrices. Consequently, this means that only a fraction of parameters is needed in comparison with classical CNN architectures to obtain similar reconstruction results. We believe that the presented framework and the proposed network architectures can be useful for learning physical models in the context of imaging and inverse problems, especially where a physical interpretation of the learned features is crucial to establish confidence in the imaging task.

Acknowledgements Open access funding provided by University of Oulu including Oulu University Hospital. We thank Jonas Adler and Sebastian Lunz for valuable discussions and comments.

References

1. Abadi, M., et al.: TensorFlow: large-scale machine learning on heterogeneous systems. Software available from https://www.tensorflow.org/ (2015)
2. Adler, J., Öktem, O.: Solving ill-posed inverse problems using iterative deep neural networks. Inverse Prob. **33**(12), 124007 (2017)
3. Antholzer, S., Haltmeier, M., Schwab, J.: Deep learning for photoacoustic tomography from sparse data. Inverse Probl. Sci. Eng. **27**, 987–1005 (2019)
4. Bergerhoff, L., Cárdenas, M., Weickert, J., Welk, M.: Stable backward diffusion models that minimise convex energies. ArXiv preprint arXiv:1903.03491 (2019)
5. Calvetti, D., Somersalo, E.: Hypermodels in the Bayesian imaging framework. Inverse Probl. **24**, 034013 (2008)
6. Chen, Y., Yu, W., Pock, T.: On learning optimized reaction diffusion processes for effective image restoration. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5261–5269 (2015)
7. Coates, A., Ng, A., Lee, H.: An analysis of single-layer networks in unsupervised feature learning. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, pp. 215–223 (2011)
8. Douiri, A., Schweiger, M., Riley, J., Arridge, S.: Local diffusion regularization method for optical tomography reconstruction by using robust statistics. Opt. Lett. **30**(18), 2439–2441 (2005)
9. Ehrhardt, M.J., Arridge, S.R.: Vector-valued image processing by parallel level sets. IEEE Trans. Image Process. **23**(1), 9–18 (2013)
10. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press. http://www.deeplearningbook.org (2016)
11. Haber, E., Ruthotto, L.: Stable architectures for deep neural networks. Inverse Probl. **34**(1), 014004 (2017)
12. Hamilton, S.J., Hauptmann, A.: Deep d-bar: real time electrical impedance tomography imaging with deep neural networks. IEEE Trans. Med. Imaging **37**, 2367–2377 (2018)
13. Hamilton, S.J., Hauptmann, A., Siltanen, S.: A data-driven edge-preserving D-bar method for electrical impedance tomography. Inverse Probl. Imaging **8**(4), 1053–1072 (2014)
14. Hammernik, K., Klatzer, T., Kobler, E., Recht, M.P., Sodickson, D.K., Pock, T., Knoll, F.: Learning a variational network for reconstruction of accelerated MRI data. Magn. Reson. Med. **79**(6), 3055–3071 (2018)
15. Hannukainen, A., Harhanen, L., Hyvönen, N., Majander, H.: Edge-promoting reconstruction of absorption and diffusivity in optical tomography. Inverse Probl. **32**(1), 015008 (2015)
16. Hauptmann, A., Arridge, S., Lucka, F., Muthurangu, V., Steeden, J.: Real-time cardiovascular MR with spatio-temporal artifact suppression using deep learning-proof of concept in congenital heart disease. Magn. Reson. Med. **81**, 1143–1156 (2019)
17. Hauptmann, A., Lucka, F., Betcke, M., Huynh, N., Adler, J., Cox, B., Beard, P., Ourselin, S., Arridge, S.: Model-based learning for accelerated, limited-view 3-d photoacoustic tomography. IEEE Trans. Med. Imaging **37**(6), 1382–1393 (2018)
18. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
19. Helin, T., Lassas, M.: Hierarchical models in statistical inverse problems and the Mumford–Shah functional. Inverse Probl. **27**(1), 015008 (2010)
20. Hofer, C., Kwitt, R., Niethammer, M., Uhl, A.: Deep learning with topological signatures. In: Advances in Neural Information Processing Systems, pp. 1634–1644 (2017)
21. Jin, K.H., McCann, M.T., Froustey, E., Unser, M.: Deep convolutional neural network for inverse problems in imaging. IEEE Trans. Image Process. **26**(9), 4509–4522 (2017)
22. Kang, E., Min, J., Ye, J.C.: A deep convolutional neural network using directional wavelets for low-dose X-ray CT reconstruction. Med. Phys. **44**(10), e360–e375 (2017)
23. Khoo, Y., Lu, J., Ying, L.: Solving parametric PDE problems with artificial neural networks. arxiv:1707.03351v2 (2017)
24. Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., et al.: Interpretability beyond feature attribution: quantitative testing with concept activation vectors (tcav). In: International Conference on Machine Learning, pp. 2673–2682 (2018)
25. Kimmel, R.: Numerical Geometry of Images: Theory, Algorithms, and Applications. Springer, Berlin (2003)
26. Liu, R., Lin, Z., Zhang, W., Su, Z.: Learning pdes for image restoration via optimal control. In: European Conference on Computer Vision, pp. 115–128. Springer (2010)
27. Long, Z., Lu, Y., Dong, B.: Pde-net 2.0: learning pdes from data with a numeric-symbolic hybrid deep network. ArXiv preprint arXiv:1812.04426 (2018)
28. Long, Z., Lu, Y., Ma, X., Dong, B.: Pde-net: Learning pdes from data. In: Proceedings of the 35th International Conference on Machine Learning (ICML 2018) (2018)
29. Meinhardt, T., Moeller, M., Hazirbad, C., Cremers, D.: Learning proximal operators: using denoising networks for regularizing inverse imaging problems. In: International Conference on Computer Vision, pp. 1781–1790 (2017)
30. Perona, P., Malik, J.: Scale-space and edge detection using anisotropic diffusion. IEEE Trans. Pattern Anal. Mach. Intell. **12**(7), 629–639 (1990)
31. Raissi, M., Karniadakis, G.E.: Hidden physics models: machine learning of nonlinear partial differential equations. arxiv:1708.00588v2 (2017)
32. Ronneberger, O., Fischer, P., Brox, T.: U-net: convolutional networks for biomedical image segmentation. In: International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 234–241. Springer (2015)
33. Ruthotto, L., Haber, E.: Deep neural networks motivated by partial differential equations. ArXiv preprint arXiv:1804.04272 (2018)
34. Sapiro, G.: Geometric Partial Differential Equations and Image Analysis. Cambridge University Press, Cambridge (2006)
35. Shalev-Shwartz, S., Ben-David, S.: Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press, Cambridge (2014)
36. Sirignano, J., Spiliopoulos, K.: DGM: A deep learning algorithm for solving partial differential equations. arxiv:1708.07469v1 (2017)
37. Tompson, J., Schlachter, K., Sprechmann, P., Perlin, K.: Accelerating Eulerian fluid simulation with convolutional networks. arxiv:1607.03597v6 (2017)
38. Weickert, J.: Anisotropic Diffusion in Image Processing. Teubner, Stuttgart (1998)
39. Weickert, J., Romeny, B.T.H., Viergever, M.A.: Efficient and reliable schemes for nonlinear diffusion filtering. IEEE Trans. Image Process. **7**(3), 398–410 (1998)
40. Weinan, E., Jiequn, H., Arnulf, J.: Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. arxiv:1706.04702v1 (2017)
41. Wu, Y., Zhang, P., Shen, H., , Zhai, H.: Visualizing neural network developing perturbation theory. arxiv:1802.03930v2 (2018)
42. Xu, L., Ren, J.S., Liu, C., Jia, J.: Deep convolutional neural network for image deconvolution. In: Advances in Neural Information Processing Systems, pp. 1790–1798 (2014)
43. Ye, J.C., Han, Y., Cha, E.: Deep convolutional framelets: a general deep learning framework for inverse problems. SIAM J. Imaging Sci. **11**(2), 991–1048 (2018)

44. Zhu, B., Liu, J.Z., Cauley, S.F., Rosen, B.R., Rosen, M.S.: Image reconstruction by domain-transform manifold learning. Nature **555**, 487–489 (2018)

S. Arridge received a BA Hons in Physics from Cambridge University in 1982, after which he moved to UCL Medical Physics where he completed a PhD in 1990. He subsequently joined the department of Computer Science UCL as a lecturer and has been professor of image processing since 2001 and visiting professor in the department of Mathematics since 2011. He was a founding member of the UCL Centre for Medical Image Computing in 2005 and the founder and director of the UCL Centre for Inverse Problems since 2013. He has been a member of the editorial board of the Institute of Physics journal "Inverse Problems" since 2000, and Editor-In-Chief since 2015. He is widely known as one of the originators of the field of *Diffuse Optical Tomography* (DOT). A special focus of his research in the last decade has been *Imaging from Coupled Physics* and in particular *Photoacoustic Tomography* (PAT). He has worked more generally in the application of inverse problems to other medical imaging areas including Diffusion Tensor MRI, SPECT, PET and digital tomosynthesis.

A. Hauptmann received his BSc and MSc in Mathematics from the Technical University Munich in 2011 and 2012, respectively. He received his PhD in 2017 from the University of Helsinki in Applied Mathematics. Additionally, he worked from 2013 to 2014 as R&D Scientist at the X-ray sensor manufacturer AJAT Oy in Espoo, Finland. Since 2017 he holds a position as Research Associate in Medical Image Computing at the Centre of Medical Imaging, University College London and is an Assistant Professor (tenure track) of Computational Mathematics since 2019 at the Research Unit of Mathematical Sciences, University of Oulu. His research interest is in inverse problems and tomographic imaging with a focus on medical applications, and his current focus is on combining inverse problems and data-driven methods, such as deep learning and neural networks.