Blind Image Inpainting with Sparse Directional Filter Dictionaries for Lightweight CNNs

Jenny Schmalfuss[1], Erik Scheurer[1], Heng Zhao[2], Nikolaos Karantzas[2], Andrés Bruhn[1] and Demetrio Labate[2*]

[1]Institute for Visualization and Interactive Systems (VIS), University of Stuttgart, Universitätsstraße 38, Stuttgart, 70569, Germany.
[2*]Department of Mathematics, University of Houston, 651 Phillip G Hoffman, Houston, 77204, TX, USA.

*Corresponding author(s). E-mail(s): dlabate@math.uh.edu;
Contributing authors: jenny.schmalfuss@vis.uni-stuttgart.de; erik.scheurer@simtech.uni-stuttgart.de; hzhao25@central.uh.edu; nickos@math.uh.edu; andres.bruhn@vis.uni-stuttgart.de;

Abstract

Blind inpainting algorithms based on deep learning architectures have shown a remarkable performance in recent years, typically outperforming model-based methods both in terms of image quality and run time. However, neural network strategies typically lack a theoretical explanation, which contrasts with the well-understood theory underlying model-based methods. In this work, we leverage the advantages of both approaches by integrating theoretically founded concepts from transform domain methods and sparse approximations into a CNN-based approach for blind image inpainting. To this end, we present a novel strategy to learn convolutional kernels that applies a specifically designed filter dictionary whose elements are linearly combined with trainable weights. Numerical experiments demonstrate the competitiveness of this approach. Our results show not only an improved inpainting quality compared to conventional CNNs but also significantly faster network convergence within a lightweight network design. Our code is available at https://github.com/cv-stuttgart/SDPF_Blind-Inpainting.

Keywords: Deep learning, image restoration, inpainting, neural networks, sparse representations

1 Introduction

Image inpainting is a longstanding problem in image processing, which aims to digitally remove visual corruptions from images that may be associated with scratches or other missing blocks of image information. The inpainting problem can be divided into two formulations, *blind* and *non-blind* image inpainting, depending on the amount of a-priori knowledge about the image corruption. For non-blind image inpainting, the location of the damage within the image is known and can be used within the algorithmic solution. However, in this work, we focus on the more challenging *blind inpainting* problem, which aims at recovering a missing region whose location is unknown. Such location information can be missing when random pixels of an image are damaged, or when identifying the damage would otherwise require human interaction. Due to the reduced amount of available information, the blind inpainting problem is generally more difficult to solve than non-blind inpainting.

Before the overwhelming success of neural networks for many image processing problems, the best performing strategies for blind image inpainting were model-based. They heavily relied on a mathematical framework that was instrumental to solve the problem. Due to this modeling aspect, classical image inpainting strategies (e.g., those based on variational or transform domain methods) are inherently predictable and explainable, which is often not true for learning-based approaches. In this work, we develop a strategy to combine the high accuracy and fast evaluation times of Convolutional Neural Networks (CNNs) with the interpretability of model-based ideas. Namely, we propose a novel notion of a receptive field layer that relies on the properties of a Parseval Frame dictionary (Fig. 1) specifically designed for image inpainting, which is combined with an appropriate sparsity constraint during network training. As we argue below, this new strategy brings highly desirable properties from the theory of sparse image representations into the CNN model, making it not only more lightweight but also more explainable.

1.1 Related work

In the literature, the most successful inpainting strategies can be roughly grouped into three categories:

(i) *Representation (or transform domain) methods* that formulate the inpainting problem as an optimization task in a transform domain [2, 4, 11, 13, 19, 33];

(ii) *PDE-based and Variational techniques* that recover missing data from the neighborhood points through a regularity criterion that might be associated with a PDE [3, 8, 9, 14, 32];

(iii) *Learning-based strategies* such as convolutional neural networks (CNNs) that learn an end to end mapping from input images to inpainted images based on training data [5, 10, 26, 35, 36].

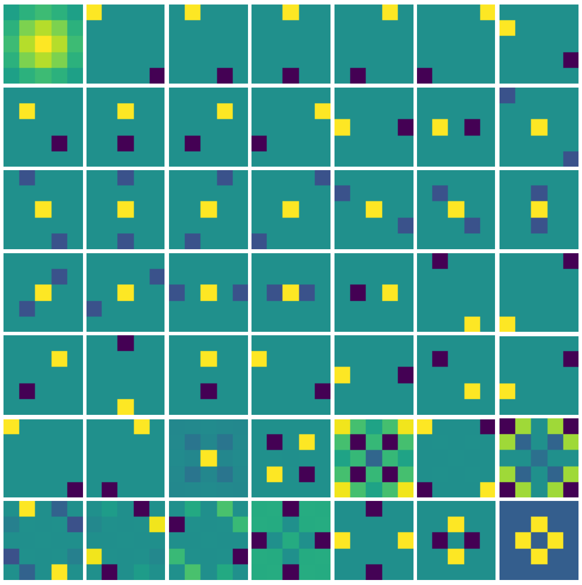


Fig. 1: The proposed 5×5 Sparse Directional Parseval Frame (SDPF) dictionary. Filters from top left to bottom right: one low-pass filter; twelve first-order finite difference filters; twelve second-order finite difference filter; 24 filters for the Parseval frame completion.

In the following, we discuss the state of the art for inpainting with (i) representation methods, (iii) learning-based strategies as well as combinations of these seemingly alternative approaches, which is also the goal of this work.

Representation methods (i) model image inpainting as a signal restoration problem, where the image is represented as a superposition of a clean component and a "noisy" one. It is then reasonable to assume that the clean component of the image has a sparse representation in some domain, e.g., in a wavelet space. That is, it can be represented using relatively few representation coefficients. Since the noise does not satisfy the same sparsity property, its energy is spread over the whole transform domain. The clean image can then be recovered by identifying its sparse representation through ℓ_1-norm minimization (in the transform domain). This intuitive argument explains the critical role of *sparsity* in image representation methods.

The study of efficient image representations has been the focus of an intense research starting with the introduction of wavelets in the late 1980's [30] and continuing with the development of more advanced multiscale representations during the following two decades (cf. the excellent review by Donoho et al. [12] about the role of such representations in image processing). Some of the most important developments in this area occurred with the introduction of curvelets [6] and shearlets [23], two multiscale methods that were shown to be provably sparser than traditional wavelets [7, 15] for a large class of images called *cartoon-like images*. The hallmark of both methods is to combine the multiresolution structure of classical wavelets with superior directional sensitivity. This directional sensitivity is achieved through the power of anisotropic scaling and the action of rotation or shear operators.

In parallel with the development of sparse representation methods, several sparsity-based algorithms for image inpainting were proposed in the literature; they include several methods based on wavelets [4, 11, 33] and shearlets [19] as well as methods such as K-SVD [2, 28, 29] that, rather than using a fixed dictionary as in the wavelet or shearlet case, build a dictionary adaptively from images. While most of these results are focused on the algorithmic side, some research also investigated performance guarantees. For instance, some results established a precise relationship between image inpainting of cartoon-like images and properties of the representation. Due to their 'geometric' properties, namely, their anisotropic support and directional sensitivity, shearlets were shown to offer a very convenient framework for inpainting as they can provably fill larger gaps than wavelets in the class of cartoon-like images [16, 19].

During the last five years or so, with the emergence of deep learning (iii) in many areas of engineering and applied mathematics, deep learning methods have gained increasing recognition also in image inpainting due to their very competitive performance. Such methods have been especially effective to address non-blind inpainting, with earlier works using simple architectures like multilayer perceptrons [22] or encoder-decoder structures [31]. Later research focused on developing alternatives for convolutions that specifically use the corruption's location, such as partial convolutions [25] or gated convolutions [39]. Most recent approaches for non-blind image inpainting use generative adversarial networks (GANs) in order to inpainting missing regions in ways that are visually hard to discriminate from similar images [37–39]. The main limitation of such methods is that their performance if highly dependent on the type of images used for training. In addition, they usually fail if the image location to be inpainted if unknown. By contrast, a much smaller number of methods were proposed to address the more challenging blind inpainting problem. Existing methods often use encoder-decoder structures based on CNN architectures [5, 10] that may contain residual blocks to improve performance [26]. The most advanced and best performing schemes in the literature for blind inpainting adopt a two-stage approach where the first stage of the algorithm estimates the location of image corruptions and the second stage applies a non-blind inpainting pass on the detected regions [35]. The improved performance comes at the cost of a significantly higher network complexity.

While deep neural networks have demonstrated impressive results and often outperform conventional methods, one major concern is the lack of interpretability due to their black-box nature [40]. Therefore, there is an increasing effort to include principles from model-based methods into deep learning for an improved interpretability. For instance, Xie et al. [36] proposed a network-based image restoration approach consisting of stacked denoising autoencoder (SDA) that takes inspirations from the K-SVD algorithm [2]. Inspired by the structure of a sparse representation method, each denoising autoencoder block is a two-layer neural network trained to reconstruct a clean image from a corrupted one with the inner layer representation constrained to be sparse. Similarly, Chaudhury and Roy [10] proposed a CNN for image restoration (IRCNN) where the hidden layers are designed to learn a data-driven sparse representation. Another method to bring principles of classical image representation into neural network designs are the Structured Receptive Field Networks (SRFN), where convolutional filters are built as linear combinations from a pre-defined dictionary, and only the coefficients of the representation are learned during training [17]. While this idea was originally proposed to learn expressive feature representations in scenarios with limited training data, we adapt and refine this idea to build our approach for image inpainting.

1.2 Contribution

In this work, we introduce a novel blind inpainting strategy that leverages the computational efficiency of a CNN along with the interpretability of mathematical representation methods. For that, we adopt the SRFN idea where each convolutional filter is a linear combination of elements from a fixed dictionary, where the coefficients of the linear combination are learned during training. Within this framework, we take advantage of the successful theory of multiscale directional representations to build a new discrete dictionary that is especially effective for image inpainting. Below are the main contributions of our approach.

(1) We design a new dictionary of filters to provide efficient representations for salient features such as edges and corners in natural images. Our filter design is based on a recently proposed mathematical framework for the construction of Parseval frames with compact support [18].
(2) We include a sparsity constraint during the training that is inspired by the sparsity norms used in model-based representation methods.
(3) We implement our inpainting strategy using a simple transform CNN architecture [10]. After examining the most effective placement of receptive field layers, we select two lightweight architectures.
(4) We run numerical experiments to demonstrate the capabilities of our method as compared to state-of-the-art methods for blind inpainting. Specifically,
    (a) we demonstrate the learning capabilities of our receptive field layers and provide an interpretation of their capabilities in terms of image representation;
    (b) we show that our filter learning strategy provides more than merely a good layer initialization and demonstrate the efficacy of our approach throughout the complete training process;
    (c) we experimentally confirm that our network strategy significantly reduces the amount of training data required for high-quality inpainting results.

We remark that another application of the SRFN idea in the context of hyperspectral classification was presented by some of the authors in [24], but with a very different rationale, network design and algorithm.

2 Method

We formulate image inpainting as an inverse problem that aims to recover an image x from its corrupted version $y = x + w$. A solution of this problem is found by solving

$$\hat{x} = \arg\min_x \|y - x\|_2^2 + \lambda \Phi(x), \qquad (1)$$

where λ is a trade-off parameter and Φ is a regularization operator enforcing some condition on the solution, e.g., sparsity.

Rather than solving Equation (1) directly through image representation methods and optimization techniques, here we opt for a learning-based approach. This choice is motivated by the efficiency of learning-based methods to address a specific corruption process, e.g., noise removal, and their fast evaluation speed compared to model-based approaches. Additionally, we include concepts from representation methods to endow our CNN approach with interpretability. In contrast to existing methods that try to combine the advantages of learning- and model-based methods by mimicking the structure representation methods at the architectural level [10, 36], our approach acts at the layer level.

2.1 Method overview

The architectural foundation for our network design is a slim, fully convolutional network architecture similar to the Image Restoration CNN (IRCNN) by Chaudhury and Roy [10]. It consists of a sequence of convolutional layers with ReLU activations, and can therefore process images of any input dimensions [27]. As shown in Fig. 2, the network architecture forms three sections that resemble the process of a classical representation method: feature extraction, nonlinear dimensionality reduction and clean image reconstruction. We select this network architecture due to this structural resemblance to representation methods as well as its simplicity. However we remark that our following strategy of expressing a convolutional filter as linear combination from a pre-designed sparse dictionary can be applied to any existing CNN architecture.

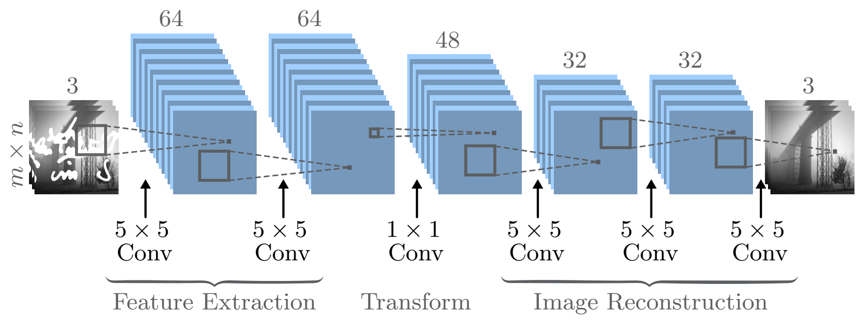


Fig. 2: Baseline network architecture, adapted from [10]. Similar to classical transform domain methods, this architecture includes 3 blocks: feature extraction, transform and reconstruction. Our geometric biased, sparse filters from Fig. 1 have the largest impact if used in the first two layers.

Following the SRFN idea, we assume that any 5×5 convolutional filter $\{S_k\}$ of our network is expressed as linear combination of 5×5 basis filters B_i, $i = 1, \dots L$ that are taken from an appropriate dictionary. Hence, any convolutional filter S_k is of the form

$$S_k = \sum_{i=1}^{L} \alpha_{k,i} B_i, \qquad (2)$$

where the filter coefficients $\alpha_{k,i}$ are learned during training. With respect to the original SRFN method [17], we introduce two innovative features that, combined with our selection of network architecture, are designed to reflect some fundamental idea from the sparsity-based approach to inpainting.

One major novelty of our approach is to build our filter dictionary as a tight frame consisting of directional filters based on the theory of shearlets. Guided by the theoretical insight that shearlet-based inpainting algorithms achieved state-of-the-art performance [19], our strategy is to select the convolutional filters of our CNN from a shearlet-like dictionary consisting of filters with high directional sensitivity. We remark that we cannot use the original shearlet filters that are defined in the Fourier domain [15, 19] nor its space-domain variants [21] due to their large support. To implement them in a CNN, we need a filter with small support. Therefore, we designed a shearlet-like filter dictionary consisting of 5×5 matrices with the following properties:

(i) it forms a Parseval frame (completeness),
(ii) it produces discrete directional differentiation in all directions (edge detection),
(iii) the filters have few non-zero entries (fast computation).

In this paper, we solve the filter design problem using the theory of *compactly supported directional framelets* [18], recently proposed by one of the authors. As shown in Fig. 1, our dictionary contains 5×5 filters with a pronounced directional response that are highly efficient to capture edges and sharp transitions in images. In the next section, we illustrate the dictionary construction in detail.

Another novelty of our approach is to impose a sparsity constraint during training that limits the number of dictionary elements allowed in any linear combination of filters. That is, in Equation (2), we only allow the sum to contain a small number of terms, e.g., three. This condition can be interpreted as a *geometric constraint*. The elements of our filter dictionary include low-pass filters and edge detectors along various discrete orientations. Therefore, a linear combination of a few filters of this type generates a kernel acting as a low-pass filter or an edge detector along selected orientations. As a consequence, while the weights of the network layer are still determined by data, our filter construction strategy results in convolutional kernels that are interpretable: They may act as direction-selective edge detectors or averaging operators. We also remark that, due to the sparsity constraint, this approach requires significantly fewer trainable weights than a standard convolution, where a weight for every filter pixel has to be learned.

In the following, we call the dictionary built for our network a *Sparse Directional Parseval Frame (SDPF) dictionary* and a convolutional layer build from this SDPF dictionary using Equation (2) (possibly with the sparsity constraint) a *SDPF constrained receptive field layer.*

2.2 Filter design

Here we adapt a method recently proposed by one of the authors in [18], to build compactly supported multidimensional wavelet-like systems with high directional sensitivity. In the 2-dimensional case, this approach starts with a compactly supported function $\phi \in L^2(\mathbb{R}^2)$ whose Fourier transform $\hat{\phi}$ satisfies the properties $\hat{\phi}(0) = 1$ and $\hat{\phi}(2\xi) = H_0(\xi)\hat{\phi}(\xi)$ for a $\mathbb{Z}^2$ periodic function $H_0(\xi) \in L^2(\mathbb{T}^2)$ called a *low-pass filter*. Then a multiwavelet system $\{\psi_i : i = 1,\dots,v\}$ is obtained by choosing *high-pass filters* $H_{1,i}(\xi) \in L^2(\mathbb{T}^2)$ such that $\hat{\psi}_i(2\xi) = H_{1,i}(\xi)\hat{\phi}(\xi)$, $i = 1,\dots,v$, and $|H_0(\xi)|^2 + \sum_{i=1}^{v}|H_{1,i}(\xi)|^2 = 1$. In practice, as the filters are expressed in the Fourier domain as trigonometric polynomials, e.g., $H_0(\xi) = \sum_{k,n=0}^{M-1} h_0(n,k)e^{2\pi i(m,k)\cdot\xi}$ for the low-pass filter, the filter construction consists in determining the matrix of *filter coefficients*, e.g., $\{h_0(m,k)\}$ for the low-pass filter coefficients. In brief, the method in [18] consists of choosing low-pass filter coefficients and an initial set of high-pass filter coefficients. Additional high-pass filter coefficients are next added until the combined set forms a frame or a Parseval frame. As we show below, this construction affords some flexibility that can be used to endow the frame with desirable properties.

We recall that a collection $\{f_i\}$ in a Hilbert space $\mathcal{H}$ is a *frame* if there are *lower* and *upper frame bounds* α, β, with $0 < \alpha \le \beta < \infty$, such that

$$\alpha \, \|f\|^2 \le \sum_i |\langle f,f_i\rangle|^2 \le \beta \, \|f\|^2 \tag{3}$$

for all elements $f \in \mathcal{H}$. A frame is a *Parseval frame* if $\alpha = \beta = 1$. The Parseval frame condition generalizes the notion of orthonormal basis and ensures that any $f \in \mathcal{H}$ can be expressed as a linear combination $f = \sum_n \alpha_i(f)\, f_i$ of frame elements. In other words, the system $\{f_i\}$ is complete in $\mathcal{H}$.

The first step in the construction of our Parseval frame is the selection of the low-pass filter coefficients. While there are several possible choices to build a compactly supported function, we choose $H_0(\xi) = \mu_0(\xi_1)\mu_0(\xi_2)$ for $\xi = (\xi_1,\xi_2) \in \mathbb{T}^2$ where

$$\mu_0(\gamma) = \left(\tfrac{1+e^{2\pi i\gamma}}{2}\right)^n, \quad \gamma \in \mathbb{T}, \tag{4}$$

is the uni-variate low-pass filter associated with the n-th order cardinal B-spline. Note that μ_0 is a trigonometric polynomial with $n+1$ terms so that the matrix of filter coefficients h_0 is a $(n+1)\times(n+1)$ matrix. By denoting as $a = (a_0,\dots,a_{N-1})$ the vectorization of the filter coefficients h_0 of H_0, we can write $H_0(\xi) = cW(\xi)$, where $N = (n+1)^2$, $c = (\sqrt{a_0},\sqrt{a_1},\dots,\sqrt{a_{N-1}})$, and $W(\xi) = (\sqrt{a_0},\sqrt{a_1}e^{2\pi i\xi},\dots,\sqrt{a_{N-1}}e^{2\pi i(N-1)\xi})^T$. We next apply the following theorem from [18]:

Theorem 1 *With the notation introduced above for N, c and W, let $H_0(\xi) = cW(\xi)$ be the low-pass filter obtained as the tensor product of two uni-variate low-pass filters associated with the n-th order cardinal B-spline (4). Suppose that Y is a $v \times N$ real-valued matrix with $v \ge \max\{N, 2^n - 1\}$ such that the rows of $\binom{c}{Y}$ form a Parseval frame in $\mathbb{R}^N$ and all rows of Y are perpendicular to c. Then the rows of the $v \times N$ matrix $B = Y diag(c)$ hold the high-pass filter coefficients inducing a Parseval frame of $L^2(\mathbb{R}^N)$.*

We will apply Theorem 1 with $n = 4$ to build a set of filter coefficients forming a Parseval frame in $\mathbb{R}^{25}$ ($N = 25$). Note that, in this case, (4) is

$$\mu_0(\gamma) = \left(\tfrac{1+e^{2\pi i\gamma}}{2}\right)^4 \tag{5}$$

so that the matrix of low-pass filter coefficients is

$$h_0 = \frac{1}{64}\begin{pmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix} \tag{6}$$

and the vector $c \in \mathbb{R}^{25}$ is obtained by vectorizing h_0 and taking the square root of the entries.

According to Theorem 1, each set of (vectorized) high-pass filter coefficients occupies a single row in a $v \times 25$ matrix Y and we can hand pick any filters that we wish, provided that each row of Y is perpendicular to c. We can exploit the flexibility afforded by this construction to enforce useful properties such as directional sensitivity at specific orientations, similar to the properties of the shearlet representation [16]. Hence, in our construction, we choose a set of first-order central difference filter coefficients h_i, $i = 1,\dots,12$, oriented at all possible discrete orientations on the 5×5 grid

$$h_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \end{pmatrix}, \qquad h_2 = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \end{pmatrix},$$
$$\dots \qquad \dots$$
$$h_{11} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \qquad h_{12} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Additionally, we select a set of second-order central difference filters h_i, $i = 13,\dots,24$,

$$h_{13} = \begin{pmatrix} 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad h_{14} = \begin{pmatrix} 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \end{pmatrix},$$
$$\dots \qquad \dots$$
$$h_{23} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 2 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad h_{24} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Next, we build a matrix $Y = \begin{pmatrix} D_1(\lambda^*) \\ D_2 \end{pmatrix}$ where $D_1(\lambda^*)$ includes our chosen first- and second order filter coefficients up to rescaling by λ^* and D_2 is a suitable completion matrix ensuring that $B = Y\,diag(c)$ holds the high-pass filter coefficients of a Parseval frame of $\mathbb{R}^{25}$. By Lemma 3.1 in [18], such a completion matrix D_2 is found if the singular values s_ν of $Q = \begin{pmatrix} c \\ D_1(\lambda^*) \end{pmatrix}$ satisfy $s_\nu \leq 1$.

To this end, we first need to define $D_1(\lambda)$ and then find $\lambda = \lambda^*$ such that Q satisfies $s_\nu \leq 1$. We convert the matrix filters h_i, $i = 1,\dots,24$, into vectors using the map $\Lambda : \mathbb{R}^{5\times 5} \to \mathbb{R}^{25}$ where

$$\Lambda(h_i) = (h_i^{5,1},\dots,h_i^{5,5},\dots,h_i^{1,1},\dots,h_i^{1,5}) \tag{7}$$

and, for a real variable λ, we let

$$d\,(\lambda, h_i) := \lambda \left(\frac{\Lambda(h_i)_k}{c_k}\right)_{k=1}^{25}. \tag{8}$$

Given $c = (c_k)_{k=1}^{25}$, and assuming an element-wise division in the definition of $d(\lambda, h_i)$, we define the matrix $D_1(\lambda) \in \mathbb{R}^{24\times 25}$ as the matrix whose rows are the $d(\lambda, h_i)$ vectors associated with the 24 high-pass filters h_i given above.

To find the value $\lambda = \lambda^*$ such that the singular values s_ν of $Q = \begin{pmatrix} c \\ D_1(\lambda^*) \end{pmatrix}$ satisfy $s_\nu \leq 1$, we solve

$$\max \quad \operatorname{trace}(c^\top c + D_1(\lambda)^\top D_1(\lambda)) \tag{9}$$

$$\text{s.t.} \quad \left\| c^\top c + D_1(\lambda)^\top D_1(\lambda) \right\| \leq 1. \tag{10}$$

Hence, by Lemma 3.1 in [18], the condition above ensures that we can find a completion matrix D_2 for which rows of

$$\begin{pmatrix} c \\ Y \end{pmatrix} = \begin{pmatrix} c \\ D_1(\lambda^*) \\ D_2 \end{pmatrix} \in \mathbb{R}^{(\upsilon+1)\times 25}, \quad \upsilon \geq 24 \tag{11}$$

form a Parseval frame for $\mathbb{R}^{25}$. Conveniently, the proof of Lemma 3.1 in [18] is constructive and provides instructions on how to build D_2. First, we perform a Singular Value Decomposition (SVD) on Q, which gives $Q = U\Sigma_1 V^\top$ with $U \in \mathbb{R}^{25\times 25}$, $V \in \mathbb{R}^{25\times 25}$ and

$$\Sigma_1 = \operatorname{diag}(\sigma_1,\dots,\sigma_{25}) \in \mathbb{R}^{25\times 25}. \tag{12}$$

Given the singular values $\sigma_1,\dots,\sigma_{25}$ and the matrix V, the completion matrix D_2 is constructed as $D_2 = \Sigma_2 V^\top \in \mathbb{R}^{25\times 25}$ with

$$\Sigma_2 = \operatorname{diag}(\sqrt{1-\sigma_1^2},\dots,\sqrt{1-\sigma_{25}^2}) \in \mathbb{R}^{25\times 25}. \tag{13}$$

From the result of the SVD, we have $\sigma_1 = 1$, then $\sqrt{1-\sigma_1^2} = 0$ and the matrix Σ_2 reduces to

$$\Sigma_2 = \operatorname{diag}(0,\sqrt{1-\sigma_2^2},\dots,\sqrt{1-\sigma_{25}^2}) \in \mathbb{R}^{25\times 25} \tag{14}$$

so that the first row of D_2 will be a zero vector. Therefore, we get 24 new filters to complete the Parseval frame for $\mathbb{R}^{25}$, and we obtain a high-pass filter matrix as

$$B = \begin{pmatrix} D_1(\lambda^*) \\ D_2 \end{pmatrix} \operatorname{diag}(c).$$

Figure 1 shows that our 5×5 filter dictionary includes a low-pass filter, 24 first- and second-order finite difference filters, followed by 24 additional filters required to obtain a Parseval frame.

The same procedure described above can be used to build filters of any support size, e.g., 3×3 or 7×7. In this work, we selected the size 5×5 since it offers a good compromise between efficiency and complexity. Filters with shorter support (e.g., 3×3) have reduced geometric sensitivity (e.g., we can only handle 4 orientations in the 3×3 grid). Filters with larger support are less local and more complex (more orientations and more elements). The filter construction in the general case is summarized in Algorithm 1.

Algorithm 1 SDPF Dictionary Construction

```
Require: The filter size $N$, a low-pass filter $h_0$ and $L$ high-pass filters $h_1 \dots h_L$ with $L{+}1 \le N$

function SDPF($N, h_0, h_1, \dots, h_L, \epsilon$)
    $a \leftarrow \Lambda(h_0) \in \mathbb{R}^N$                ▷ $\Lambda$: filter to vector
    $c \leftarrow (\sqrt{a_0}, \sqrt{a_1}, \dots, \sqrt{a_{N-1}}) \in \mathbb{R}^N$
    $\lambda \leftarrow (1, \dots, 1) \in \mathbb{R}^N$                ▷ Initialize $\lambda$
    for $i = 1 \dots L$ do
        $d_i(\lambda) \leftarrow \lambda_i \left(\frac{\Lambda(h_i)_1}{c_1}, \dots, \frac{\Lambda(h_i)_N}{c_N}\right)$
    end for
    $D_1(\lambda) \leftarrow \begin{pmatrix} d_1(\lambda) \\ \vdots \\ d_L(\lambda) \end{pmatrix} \in \mathbb{R}^{L \times N}$
    $\lambda^* \leftarrow \underset{\lambda}{\operatorname{argmax}} \ \operatorname{trace}(c^\top c + D_1(\lambda)^\top D_1(\lambda))$
        s.t. $\|c^\top c + D_1(\lambda)^\top D_1(\lambda)\| \le 1$
    $Q \leftarrow \begin{pmatrix} c \\ D_1(\lambda^*) \end{pmatrix} \in \mathbb{R}^{(L+1) \times N}$
    $V, \Sigma_1 \leftarrow$ SVD of $Q = U \Sigma_1 V^\top$
    for $i = 1 \dots L+1$ do
        $s_i \leftarrow \sqrt{1 - (\Sigma_1)_{i,i}^2}$
        if $s_i < \epsilon$ then
            $s_i \leftarrow 0$
        end if
    end for
    $\Sigma_2 \leftarrow \operatorname{diag}(s_1, \dots, s_{L+1}, 1, \dots, 1) \in \mathbb{R}^{N \times N}$
    $D_2 \leftarrow \Sigma_2 V^\top \in \mathbb{R}^{N \times N}$
    $B \leftarrow \begin{pmatrix} D_1(\lambda^*) \\ D_2 \end{pmatrix} \operatorname{diag}(c)$, eliminate 0-rows
    $P \leftarrow \begin{pmatrix} c \\ B \end{pmatrix} \in \mathbb{R}^{(v+1) \times N}$,    $v \ge N - 1$
    return $P$
end function

Ensures: Parseval filter bank $P \in \mathbb{R}^{(v+1) \times N}$, with $v$ high-pass filters and 1 low-pass filter, filters obtained by reshaping the columns of $P$ into $N \times N$ matrices.
```

3 Results

The experimental evaluation of our image inpainting algorithm is divided into four parts. In the first part, we discuss how to select the best configuration of SPDF constrained receptive field layers in our network for image inpainting. The second part provides a deeper analysis of the learned filters, and the impact of SPDF constrained receptive field layers on the training process. In the third part, we analyze the amount of training data that is required to effectively train a network with SPDF constrained receptive field layers as compared to a conventional CNN. Lastly we measure the inpainting quality and run time of our top performing network configurations in comparison to state-of-the-art inpainting methods. Preceding the experimental analysis we describe the imaging dataset used for our experiments.

3.1 Experimental data set

The dataset used for our experiments consists of 225,100 images with 256 × 256 pixels from the *Places* data set [41], which we overlaid the images with handwriting masks (line width about 10 pixels) extracted from scanned pages, cf. Fig. 8 for examples. The full dataset is available at https://doi.org/10.18419/darus-2886. The masks are either colored white, in a random color or with Gaussian noise ($\mu = 0.4363$, $\sigma = 0.2737$, according to color distribution of the selected images from Places). Note that we obtained our mask library by rotating a set of 56,275 handwriting masks by 0°, 90°, 180° and 270°, resulting in a total of 225,100 masks.

Our handwriting masks provide a variable coverage of a given image, ranging from 1% to 25% of the total number of pixels in an image. Table 1 lists the distribution of masks by the size of the area they occlude, and how they are split into training and test sets. Note that not enough samples were available in the coverage range 20-25%, to allow an even distribution of test samples.

Table 1: Number of handwriting images in the training and test sets, grouped by occlusion area.

Occlusion	Training	Test
0-5%	100,000	1,000
5-10%	100,000	1,000
10-15%	10,000	1,000
15-20%	10,000	1,000
20-25%	1,000	100

Table 2: Inpainting comparison for network configurations that use exactly one SDPF constrained receptive field layer. Performance values are computed on the test set, after the network was trained on 5,000 training images for 100 epochs; results are averaged over 10 training runs.

Configuration	**Param.**	**MSE** ($\times 10^{-2}$)	**L$_1$** ($\times 10^{-2}$)	**PSNR**	**SSIM**
`C-C-c-C-C-C` (IRCNN)	172,113	0.1437	1.3949	30.5622	0.9437
`B-C-c-C-C-C` (GBCNN)	170,705	**0.1074**	**0.9458**	**32.2214**	**0.9552**
`C-B-c-C-C-C`	**82,001**	0.1180	1.0838	31.7225	0.9520
`C-C-c-B-C-C`	138,321	0.1283	1.2192	31.1119	0.9469
`C-C-c-C-B-C`	149,585	0.1359	1.2727	30.8522	0.9430
`C-C-c-C-C-B`	171,409	0.1366	1.3159	30.7806	0.9442

3.2 Network configuration and training

To identify the best network configuration for inpainting, we systematically tested different architectures where any convolutional layer is either a SDPF constrained receptive field layer or a conventional convolutional layer. In the first case, any convolutional kernel is a linear combination of only 3 filters that are randomly selected out of the 25 filters of the SPDF dictionary. The number 3 indicates that we imposed a sparsity constraint with sparsity 3. In our implementation, the sparsity value s is treated as a hyperparameter of the neural network indicating that s filters are randomly chosen from the SPDF dictionary to generate each convolutional filter when the network is initialized. We tested several values of the sparsity level s and heuristically found that sparsity $s = 3$ is the best choice for our dataset. A higher sparsity level would increase the number of trainable parameters without improving or even downgrading performance.

We trained the networks on a reduced training set with white masks, consisting of 5,000 images (1,000 per occlusion area range) to limit the computational budget. The CNNs were implemented using Tensorflow [1] with the Adam optimizer [20] (using default setting) and trained for 100 epochs with batch size 10. Our code is available at https://github.com/cv-stuttgart/SDPF_Blind-Inpainting.

To describe the network architectures, in the following we denote a conventional 5×5 convolutional layer by `C`, while we denote by `c` a conventional 1×1 convolutional layer. By contrast, we denote by `B` a SDPF constrained receptive field layer with sparsity 3. As our dictionary is constructed from 5×5 filters, the third layer `c` is never replaced by a receptive field layer. Note that the configuration with only convolutional layers (`C-C-c-C-C-C`) is the IRCNN from [10].

Table 2 reports the image inpainting performance for multiple network configurations that are derived from the CNN architecture in Fig. 2 by using either conventional convolutional layers or SDPF constrained receptive field layers. We note that all configurations with a receptive field layer (`B`) outperform the original IRCNN architecture for all measured metrics, indicating the improvement due to our modified design and despite the reduced number of trainable parameters.

Among the various configurations, the best inpainting quality is achieved with the first layer being a SDPF constrained receptive field layer and the remaining layers being conventional convolutional layers (`B-C-c-C-C-C`). For brevity, we call this configuration *Geometric-Biased CNN* (GBCNN), due to the geometric bias (high directional sensitivity) that is imposed by the combination of SDPF filters and sparsity constrained in the first layer. We explain the excellent performance of this configuration with the improved ability of the SDPF constrained receptive field layer to respond to geometric cues in images, such as edges or corners - an ability which is particularly effective in the first layer(s). This is supported the performance measures in Table 2, which show a decreasing inpainting performance with the depth of the SDPF filters in the network. Out of all network configurations that use SDPF filters, `C-C-c-C-C-B` yields the worst performance,

Table 3: Inpainting comparison of network configurations based on the GBCNN architecture, trained on 5,000 images for 100 epochs with results averaged over 10 training runs. Best results displayed in bold.

Configuration	**Param.**	**MSE** ($\times 10^{-2}$)	**L$_1$** ($\times 10^{-2}$)	**PSNR**	**SSIM**
`B-C-c-C-C-C` (GBCNN)	170,705	**0.1074**	**0.9459**	**32.2213**	**0.9552**
`B-C-c-C-C-B`	170,001	0.1113	**0.9867**	**32.0204**	0.9532
`B-C-c-C-B-C`	148,177	0.1138	1.0383	31.8524	0.9519
`B-C-c-C-B-B`	147,473	0.1230	1.0890	31.5003	0.9487
`B-C-c-B-C-C`	136,913	0.1160	1.0469	31.7855	0.9519
`B-C-c-B-C-B`	136,209	0.1183	1.0465	31.6908	0.9505
`B-C-c-B-B-C`	114,385	0.1255	1.1303	31.3345	0.9479
`B-C-c-B-B-B`	113,681	0.1441	1.2946	30.5393	0.9405
`B-B-c-C-C-C` (GBCNN-L)	80,593	**0.1109**	1.0113	31.9925	**0.9538**
`B-B-c-C-C-B`	79,889	0.1166	1.0451	31.7482	0.9513
`B-B-c-C-B-C`	58,065	0.1391	1.3125	30.7628	0.9430
`B-B-c-C-B-B`	57,361	0.1294	1.1457	31.1907	0.9462
`B-B-c-B-C-C`	46,801	0.1184	1.0562	31.6847	0.9508
`B-B-c-B-C-B`	46,097	0.1244	1.0977	31.4052	0.9484
`B-B-c-B-B-C`	24,273	0.1292	1.1401	31.1820	0.9462
`B-B-c-B-B-B`	**23,569**	0.1484	1.3634	30.3444	0.9378

even though it still outperforms the fully convolutional IRCNN architecture. Also, the weakening effect of the SDPF filters with increasing network depth appears to be independent of the network's parameter count (and hence its expressiveness), indicating that the improved performance indeed comes from the introduction of model assumptions into the network.

Table 3 lists an extended architectural evaluation based on the top performing GBCNN configuration (`B-C-c-C-C-C`). It reports the performance of the network configurations resulting from all possible combinations of receptive field and convolutional layers following the first layer.

We observe that the modification of the second layer has the largest impact on the amount of trainable parameters. When using a second layer with SDPF filters and sparsity constraint (`B-B-c-C-C-C`), the amount of parameters is more than halved compared to using a fully convolutional one. In accordance with the results from Table 2, receptive field layers impair the inpainting performance when placed in the image reconstruction section of the network (cf. Fig. 2, layers four to six). Out of those three positions, using SDPF filters in the last layer leads to the least degradation of results.

This yields two candidates with fewer parameters and only a slight degradation in performance compared to GBCNN: the configurations `B-C-c-C-C-B` with 170,001 parameters and `B-B-c-C-C-C` with 80,593 parameters. We remark that PSNR and MSE do not behave necessarily the same for each architectures due to averaging the numbers per image, and subsequently per dataset. We then select `B-B-c-C-C-C` as the lightweight version of GBCNN and denote it as GBCNN-L. Below, we will compare both network configurations against state-of-the-art methods.

3.3 Filter analysis

Here we analyze the properties of SDPF filters in the first network layer. We compare the filter structure to fully convolutional filters after a completed training, and investigate their influence on the training process.

3.3.1 SDPF filter response

To analyze the differences between convolutional and SDPF constrained receptive field layers, we investigate the structure of learned features when they are applied to natural images. Fig. 3 visualizes the 64 filters in the first layer of IRCNN

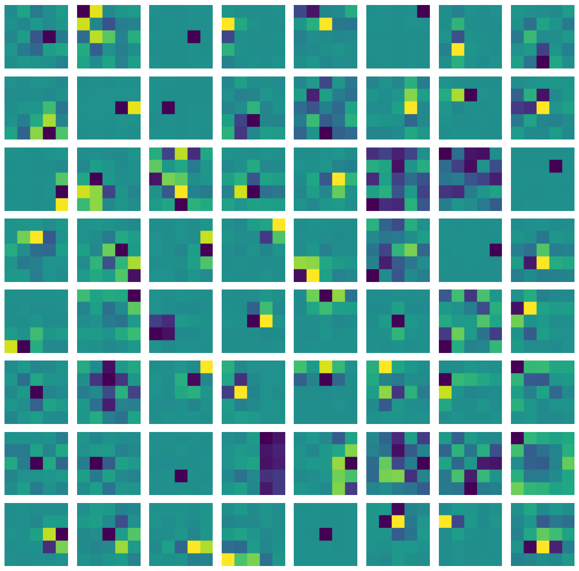
(3a) and GBCNN (3b) after being trained for 100 epochs on the full test set of 221,000 images with white masks.

Filters learned in the first layer of the GBCNN are visually more structured and include several first- and second order difference filters. These clearly result from the linear combinations of a few elements from the SDPF dictionary. We also notice a few low-pass filters (e.g., row 2, column 7 and row 8, column 8). By contrast, the IRCNN filters appear less symmetrical and include almost no elements identifiable as difference filters.

In Fig. 4, we illustrate the responses of a representative selection of filters applied to natural images, to highlight the different filter characteristics. To this end, we visualize the effect of randomly selected filters from the first layers of IRCNN and GBCNN on images from the test data set. The filters learned in the SDPF constrained receptive field layers in Fig. 4b act typically as edge detectors along selected orientations (columns 1,3,5,6) but also include a low-pass (column 2) and high-pass (column 4) filter. By contrast, the filter responses for the IRCNN architecture in Fig. 4a are less structured and do not include strong directional responses. This highlights the geometric character and interpretability of the filters associated with our SDPF constrained receptive field layers that are inspired by principles of sparse image approximations.

3.3.2 SDPF filters during training

Here we investigate whether applying our strategy based on pre-designed filters is advantageous throughout the whole training process. One could suppose that these filters simply act as a good "layer initialization" during the first epochs; one could then lift the SDPF dictionary constraint after a certain number of epochs after which the network would convergence to an even better model. To investigate this possibility, we implemented a version of a receptive field layer that we trained for a fixed number of epochs under the constraint that convolutional kernels are taken as linear combinations our SDPF dictionary elements; after a prescribed number of epochs, we used the learned filters as initial weights for a conventional convolutional layer that we trained further.


(a) Conventional convolutional kernels (layer type `C`)

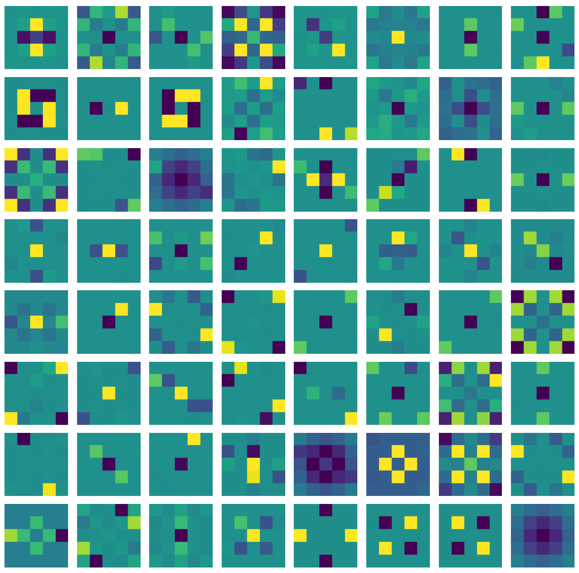

(b) Kernels in receptive field layer (layer type `B`)

Fig. 3: The 64 learned filters in the first layer of the CNN architecture in Fig. 2 resulting from different types of training strategies.

Fig. 5 shows the result of this numerical experiment, where we trained a GBCNN for 60 epochs, after which we lifted the SDPF constraint. This effectively replaces the SDPF constrained receptive layer with a conventional convolutional layer that we initialized with the filters learned after 60

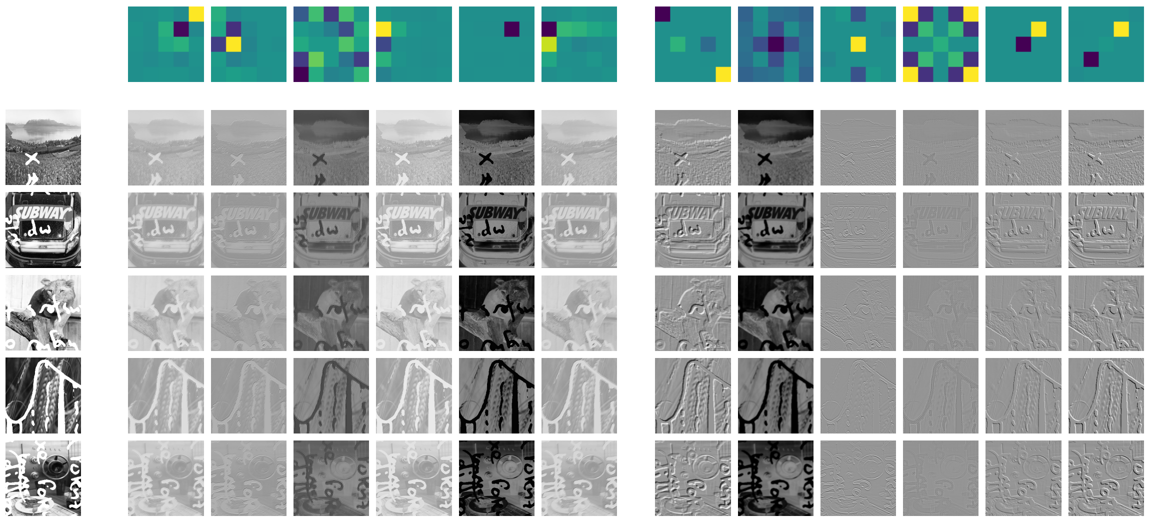


Fig. 4: Filter responses on five natural images with overlaid handwriting for six randomly sampled filters from the first layer of the IRCNN and GBCNN networks (cf. Fig. 3 for entire filter sets). The filters are displayed in the top row and the original images are shown in the first column on the right.

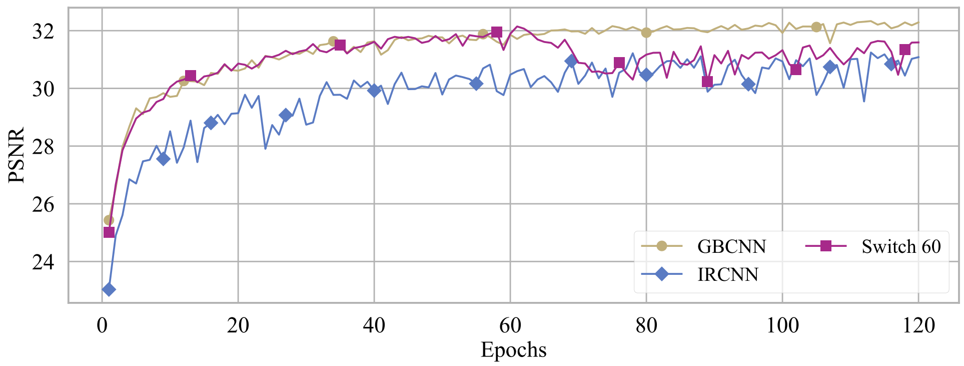


Fig. 5: Inpainting performance as a function of the number of training epochs. The GBCNN performance is always above the IRCNN performance. The purple curve (labeled Switch 60) shows the effect of relaxing the SDPF constrained receptive field layer into a conventional convolutional layer after 60 epochs; the network performance decreases to a level comparable to the IRCNN. Results are averaged over 10 runs.

epochs. The figure compares the inpainting performance of the model resulting from this experiment against a GBCNN and an IRCNN. Remarkably, despite the good initialization after 60 epochs, the performance of the network trained with conventional convolutional layers decreases with respect to the GBCNN and falls to the level of a network that used convolutions during the whole training.

To further illustrate this phenomenon, Fig. 6 visualizes a randomly selected set of filters over the duration between lifting the SDPF constraint until the training ends. We see that the filter responses change dramatically, even though the filters appear to change very gradually. In some instances, the filter responses lose their sensitivity to edges and salient features that is common for filters obtained from the SDPF dictionary (cf. Fig. 4a)

All in all, these observations show that our training strategy based on SPDF dictionary and

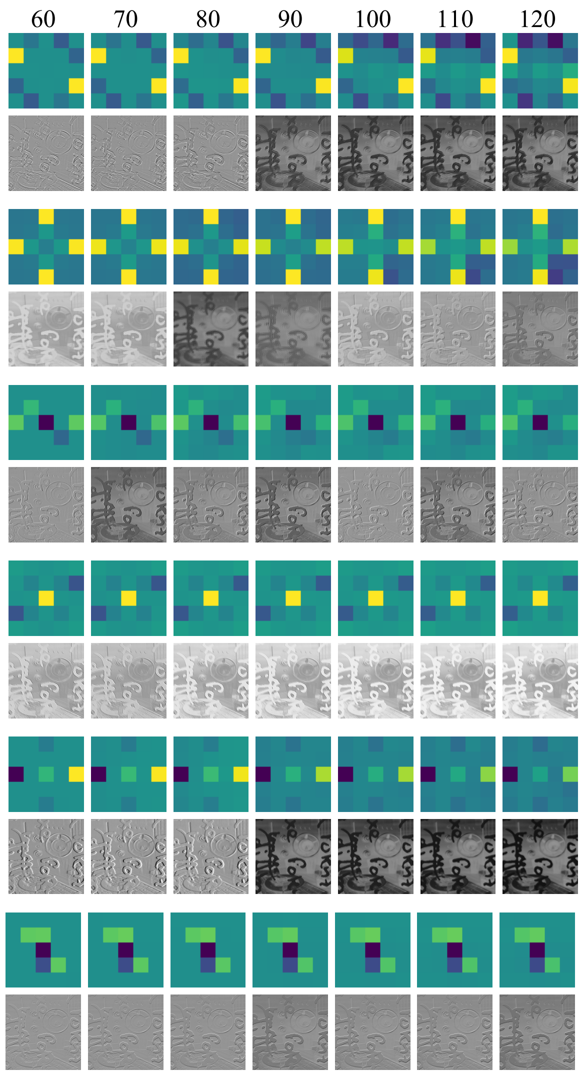


Fig. 6: Evolution of selected kernels in the first network layer, after the receptive field layer is relaxed into a conventional one. Each kernel is displayed above its response every 10 epochs, between 60-120 epochs.

sparsity constraint affects the convergence of the network kernels throughout the entire training process and its effect cannot be reduced to a clever filter initialization.

3.4 Inpainting performance with increasing training set size

In practical applications of neural networks, it is often important to decide the amount of training data that is required for the network to converge to a satisfactory model.

We systematically investigated the inpainting performance of our GBCNN as compared to a IRCNN for different amounts of training data with white masks and reported the results in Fig. 7. The figure shows the image inpainting performance in PSNR for images corrupted by occlusions affecting a different fraction of the image area as a function of the number of training samples (between 10 and 10,000 samples). The networks are trained for 100 epochs and the results averaged over 10 runs. Compared to IRCNN, our GBCNN approach exhibits higher PSNR values on the test set after being trained on relatively few data samples. This difference in performance reduces with more data, but remains very significant on images with smaller areas to inpaint.

Since our GBCNN uses a lower number of trainable parameters than an IRCNN (with the same architecture), it was expected that the former would converge with a lower number of training samples. This intuition was confirmed by our numerical experiments. We explain this behavior with the improved ability of the SPDF dictionary to capture the essential image characteristics which is also the reason for the competitive performance found in the shearlet-based inpainting algorithm that motivated this study. By contrast, the IRCNN requires more training samples to reach the same performance as the GBCNN.

3.5 Benchmark comparison of inpainting performance

We compared the performance of our GBCNN and GBCNN-L against state-of-the-art algorithms for blind image inpainting, including IRCNN [10] and VCNet [35]. Our comparison does not include the wide body of work on non-blind deep inpainting methods, e.g., partial convolutions [25], contextual attention [38], convolutions [39] or generative approaches, due to their use of information about the corruption's location and the resulting problem simplification. However, we included the non-blind method ShearLab [19] in the comparison, as our approach is partially motivated by sparsity-based ideas underlying this approach.

The networks are compared on all three versions of the dataset (masked with white, random

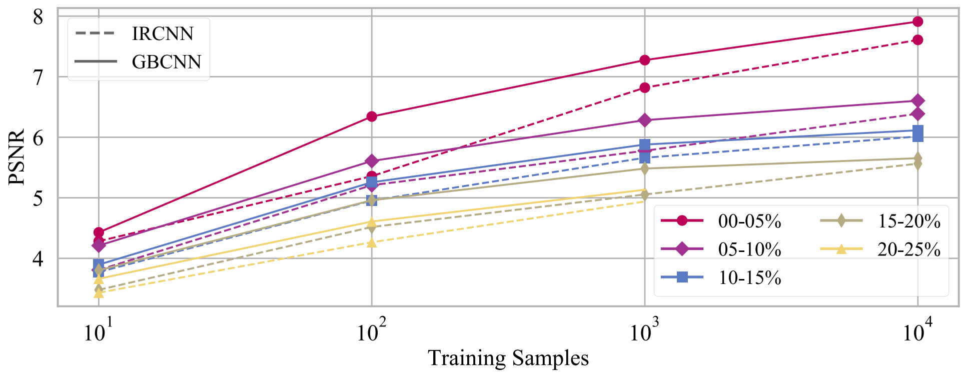


Fig. 7: Image inpainting performance in PSNR for IRCNN (dashed) and GBCNN (solid), for different occlusion areas (distinguished by color and marker) on the training images. For 20-25% coverage only 10^3 samples were available. Networks are evaluated after training for 100 epochs and the numbers are averaged over 10 network trainings.

color or Gaussian noise) and trained on the full data sets (221,000 images) for 100 epochs. VCNet, however, is trained from scratch using 160,000 iterations with a batch size of 12 images following the original paper training schedule[1]. Each network is trained five times, and the training run with the best performance is displayed.

Table 4 reports the image inpainting performance in PSNR on the full test sets and their subsets associated with different fractions of the image area lost to occlusion. Since the qualitative performance, as measured using the SSIM metric, is very similar to the PSNR, we did not report it. The results for Shearlet inpainting are only stated for the first (white masked) dataset, as Shearlab uses explicit information about the mask's location and is therefore invariant to the masking color.

The table shows the dataset type has a significant influence on the overall inpainting quality as well as the performance of the different methods. While all blind inpaintig methods (VCNet, IRCNN, GBCNN and GBCNN-L) perform well on the white-masked dataset, the more complex VCNet performs low on the more difficult randomly colored and Gaussian noise datasets. We remark that VCNet requires 1,600 times more training epochs than GBCNN and uses a complex multistep algorithm. Except for VCNet, all learning-based approaches consistently outperform the (non-blind) shearlet-based inpainting method from [19]. This is remarkable, considering that the latter method is non-blind, and confirms the superior performance of learning-based methods in inpainting.

Among the more lightweight CNN-based approaches (GBCNN-L, GBCNN and IRCNN), the differences per dataset are much smaller. Our sparse architectures GBCNN and GBCNN-L both outperform IRCNN's PSNR on the white and randomly colored datasets, with the best results for white masks achieved by GBCNN-L and for randomly colored by GBCNN. IRCNN leads on the Gaussian noise set, closely followed by GBCNN.

When we examine the performance on the subsets by percentage of masked image area, the table shows an interesting shift in the top-performing methods for the white dataset. For relatively small occlusions (0-5%), GBCNN-L outperforms the other methods, with GBCNN performing closely and VCNet showing a significantly worse performance (37.8666 vs 39.9796 dBs). For larger occlusions (5-10%) GBCNN has the best performance with GBCNN-L performing closely. For the largest tested occlusions (15-20% and 20-25%) VCNet achieves the best performance, even though the improvement with respect to GBCNN is less that 0.5 dBs. We attribute this observation to the fact that for large inpainting areas, the information about the location of the missing region becomes more important and VCNet, taking advantage of its greater complexity, is able to

[1]Retraining the network became necessary since the pretrained VCNet model provided by the authors in [35] did not achieve comparable results even after tuning it on our data set for 100,000 additional iterations.

Table 4: Image inpainting performance measured as average PSNR on the entire test set or on subsets associated with the percentage of image lost due to occlusion. Best result by column in bold, second best underlined.

Method	**Entire set**	**0–5%**	**5–10%**	**10–15%**	**15–20%**	**20–25%**
	Dataset white					
Shearlet [19]	30.7005	34.5694	31.4638	29.5394	27.7532	25.4599
VCNet [35]	32.6120	37.8666	33.1650	**30.9287**	**29.0408**	**27.0796**
IRCNN [10]	32.5554	38.9941	33.0309	30.5410	28.2957	26.1550
GBCNN (ours)	<u>32.9497</u>	<u>39.5763</u>	**33.3030**	<u>30.8849</u>	<u>28.6630</u>	<u>26.6679</u>
GBCNN-L (ours)	**33.0063**	**39.9796**	<u>33.2698</u>	30.8267	28.5877	26.6221
	Dataset random color					
VCNet [35]	28.7741	33.9609	29.6332	27.2164	24.8558	22.6557
IRCNN [10]	32.0852	37.9933	32.5079	30.2452	28.1787	26.2413
GBCNN (ours)	**32.4776**	**38.3913**	**32.9436**	**30.6258**	**28.5373**	**26.6047**
GBCNN-L (ours)	<u>32.2264</u>	<u>38.3673</u>	<u>32.5892</u>	<u>30.3000</u>	<u>28.2446</u>	<u>26.2721</u>
	Dataset Gaussian noise					
VCNet [35]	29.8666	35.7458	30.3789	27.8976	25.9770	24.0801
IRCNN [10]	**32.6010**	<u>39.5739</u>	**32.9097**	**30.3680**	**28.1927**	**26.1986**
GBCNN (ours)	<u>32.5193</u>	**39.6120**	<u>32.8116</u>	<u>30.2465</u>	<u>28.0567</u>	<u>26.0254</u>
GBCNN-L (ours)	32.1891	39.2154	32.4960	29.9362	27.7528	25.7499

recover large blocks of missing image information more effectively. Yet, the performance of GBCNN is not far off (about -0.4dBs) and it seems that VCNet's larger complexity does not help to detect non-white masks.

The comparison of the different network approaches shows that, due to their ability to capture salient image features in images, our GBCNN and GBCNN-L perform very competitively overall, even outperforming the more sophisticated VCNet approach for all mask types. Even though our approach is designed for the detection of edges and not for noise removal, also in the presence of masked filled with Gaussian-noise, the inpainting performance is still comparable to the IRCNN architecture and even delivers top results when a relatively small fraction of the area is affected by occlusion.

Representative reconstruction results for all inpainting methods and datasets are shown in Fig. 8. In terms of visual quality, close inspection shows that our approach is more effective to remove the mask contents than VCNet. We observed this property particularly in the random-colored and Gaussian-noise datasets and interpret it as a consequence of our special filter selection process that is designed to capturing edge-like structures with high efficiency.

3.5.1 Evaluation time and parameters

We also compared the inference times for all of the inpainting algorithms considered in this study in Table 5, and additionally considered the number of trainable parameters. All run times are computed using a single NVIDIA Tesla V100 PCIe graphics card with 32 GB memory. All network approaches were implemented in Python and Tensorflow, while the shearlet-based method has a Matlab implementation.

Top times are delivered by IRCNN, GBCNN and GBCNN-L with about 2.5 msec, since inference requires a computationally cheap feed forward pass on a simple network architecture. The time differences between these three methods are negligible, and might be due to small fluctuations in the time measurements. Due to the lightweight architecture resulting from the placement of receptive field layers, GBCNN-L has the lowest parameter count, followed by GBCNN and IRCNN with twice as many parameters. Compared to the three networks above, VCNet has about 20 times more parameters and takes roughly 5 times longer for

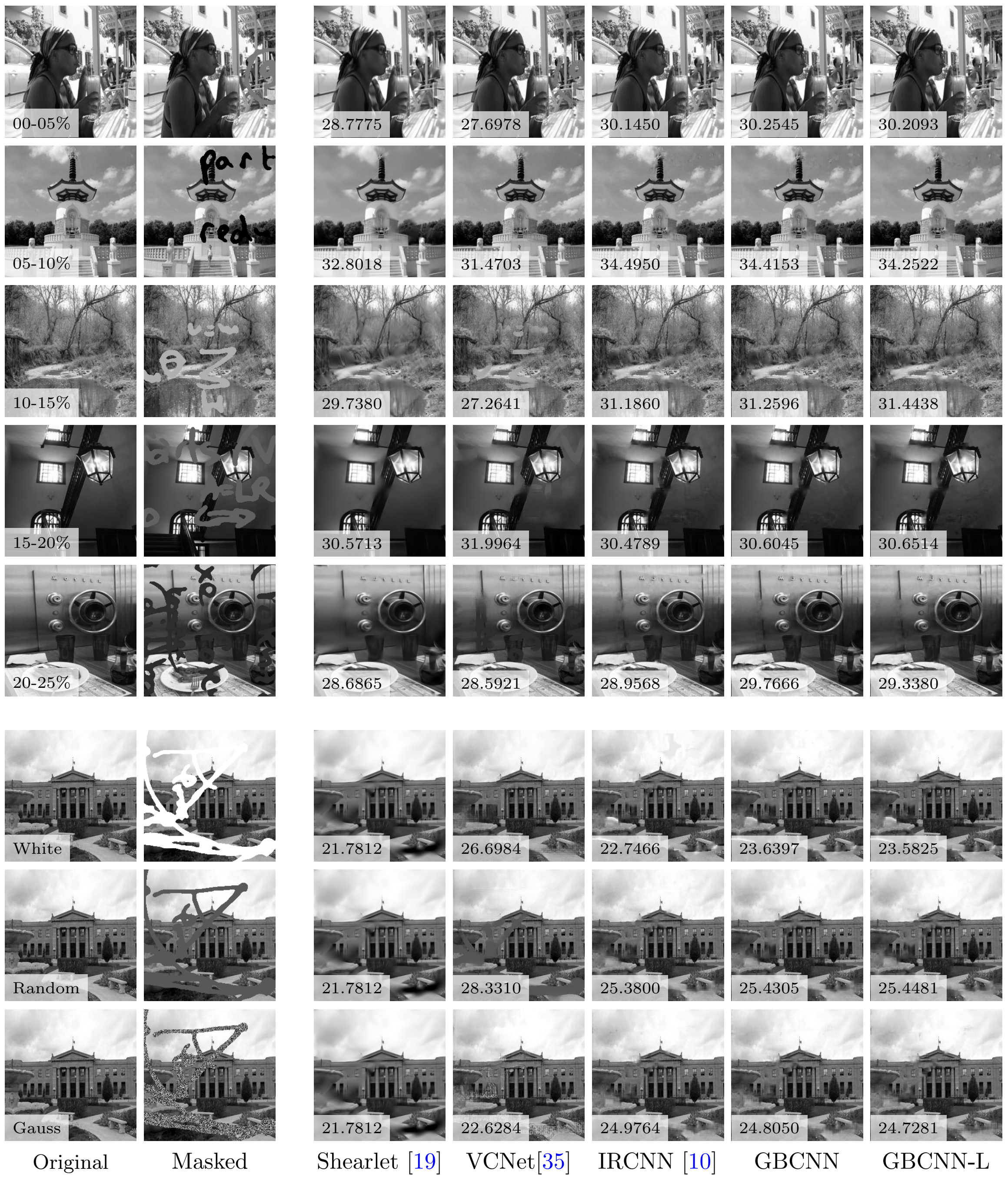


Fig. 8: Inpainting comparison on images with varying occlusion percentages on random-color dataset (top) and dataset comparison on 20-25% occlusion (bottom); PSNR per method reported in the image.

inference. This comparison shows the advantages of SDPF constrained receptive field layers for the development of lightweight and fast network architectures.

The iterative nature of the shearlet-based inpainting algorithm significantly increases its run time (about 10^4 times larger) as compared to all network implementations.

Table 5: Comparison of network parameter count and average evaluation time μ_t per image (size 256×256) in milliseconds for inpainting methods.

	ShearLab [19]	**VCNet** [35]	**IRCNN** [10]	**GBCNN**	**GBCNN-L**
Parameters	–	3,789,892	172,113	170,705	**80,593**
μ_t [msec]	29,147.001	17.460	2.539	**2.453**	2.534

4 Conclusions

We have introduced a novel strategy for blind image inpainting that brings model-based principles from the theory of sparse representation into the design of a new deep learning model. Our approach employs a specifically designed filter dictionary, called SPDF, in combination with a sparsity constraint, that is motivated by the success of shearlet representations in image processing applications. With these novel concepts we develop two lightweight network models called GBCNN and GBCNN-L.

One main advantage of our approach to blind inpainting is the increased interpretability. As compared to the conventional CNN approach where there is essentially no control on the kernel structure, the kernels learned by our GBCNN and GBCNN-L reflect the geometric properties of the SPDF dictionary that are critical to capture salient image features such as edges and corners. This behavior is consistent with the model-based principles that guided our design strategy.

By integrating model-based principles into a simple and light weight network architecture, our approach outperforms not only conventional CNN schemes with similar architectures in terms of inpainting quality. It also excels the significantly more complex VCNet algorithm, which applies a multistep strategy for blind image inpainting.

However, our approaches exhibit a reduced inpainting quality in the case where the region to be inpainted is relatively large. We believe the reduction in competitiveness of our method in this case to be explained by the support size of our filter dictionary, whose element were selected to have fixed size of 5×5 pixels. This observation suggests that our method could possibly be improved by considering a filter dictionary with elements having multiple size supports, e.g., 5×5, 7×7 and 9×9, and the idea of using filters of different support size in CNNs has been already employed successfully in the deep learning literature [34]. This extension would be fully consistent with the theoretical framework of shearlet-based inpainting [16] that inspired the present work, since the desirable properties of the shearlet representation system include not only directional sensitivity but also multiresolution.

Acknowledgments. Authors thank Prof. Dr. D. Göddeke for enabling this collaboration and his constant support and advise, Prof. Dr. B. Haasdonk and Prof. Dr. C. Rohde for providing the handwritten notes, and K. Safari for his help with the implementation. DL acknowledges support by NSF-DMS 1720487, 1720452 and HPE DSI/IT at UH. JS acknowledges partial funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - Project-ID 251654672 - TRR 161 (Project B04) and by the International Max Planck Research School for Intelligent Systems (IMPRS-IS).

References

[1] Abadi M, Barham P, Chen J, et al (2016) TensorFlow: A system for large-scale machine learning. OSDI 16:265–283. URL https://www.tensorflow.org/

[2] Aharon M, Elad M, Bruckstein A (2006) K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. IEEE Transactions on Signal Processing 54(11):4311–4322

[3] Arias P, Facciolo G, Caselles V, et al (2011) A variational framework for exemplar-based image inpainting. International Journal of Computer Vision 93(3):319–347

[4] Cai JF, Dong B, Osher S, et al (2012) Image restoration: Total variation, wavelet frames, and beyond. Journal of the American Mathematical Society 25(4):1033–1089

[5] Cai N, Su Z, Lin Z, et al (2017) Blind inpainting using the fully convolutional neural network. The Visual Computer 33(2):249–261

[6] Candès A, Donoho D (2000) Curvelets a surprisingly effective nonadaptive representation for objects with edges. In: Schumaker L (ed) Curves and Surfaces. Vanderbilt University Press, p 105–120

[7] Candès EJ, Donoho DL (2004) New tight frames of curvelets and optimal representations of objects with piecewise C2 singularities. Communications on Pure and Applied Mathematics 57(2):219–266

[8] Chan TF, Shen J (2005) Variational image inpainting. Communications on Pure and Applied Mathematics 58(5):579–619

[9] Chan TF, Shen J, Zhou HM (2006) Total variation wavelet inpainting. Journal of Mathematical Imaging and Vision 25(1):107–125

[10] Chaudhury S, Roy H (2017) Can fully convolutional networks perform well for general image restoration problems? Proc International Conference on Machine Vision Applications (MVA) pp 254–257

[11] Dong B, Ji H, Li J, et al (2012) Wavelet frame based blind image inpainting. Applied and Computational Harmonic Analysis 32(2):268–279

[12] Donoho DL, Vetterli M, DeVore RA, et al (1998) Data compression and harmonic analysis. IEEE transactions on information theory 44(6):2435–2476

[13] Elad M, Starck JL, Querre P, et al (2005) Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA). Applied and Computational Harmonic Analysis 19(3):340–358

[14] Esedoglu S, Shen J (2002) Digital inpainting based on the Mumford–Shah–Euler image model. European Journal of Applied Mathematics 13(4):353–370

[15] Guo K, Labate D (2007) Optimally sparse multidimensional representation using shearlets. SIAM Journal on Mathematical Analysis 39:298–318

[16] Guo K, Labate D, Ayllon JPR (2020) Image inpainting using sparse multiscale representations: Image recovery performance guarantees. Applied and Computational Harmonic Analysis 49(2):343–380

[17] Jacobsen JH, Van Gemert J, Lou Z, et al (2016) Structured receptive fields in CNNs. In: Proc. Conference on Computer Vision and Pattern Recognition, pp 2610–2619

[18] Karantzas N, Atreas N, Papadakis M, et al (2019) On the design of multi-dimensional compactly supported Parseval framelets with directional characteristics. Linear Algebra and its Applications 582:1–36

[19] King EJ, Kutyniok G, Zhuang X (2014) Analysis of inpainting via clustered sparsity and microlocal analysis. Journal of Mathematical Imaging and Vision 48(2):205–234

[20] Kingma DP, Ba J (2015) Adam: A method for stochastic optimization. In: Proc. International Conference on Learning Representations

[21] Kutyniok G, Lim WQ (2010) Compactly supported shearlets are optimally sparse. Journal of Approximation Theory 163:1564–1589

[22] Köhler R, Schuler C, Schölkopf B, et al (2014) Mask-specific inpainting with deep neural networks. In: Pattern Recognition, vol 8753. Springer International Publishing, p 523–534

[23] Labate D, Lim WQ, Kutyniok G, et al (2005) Sparse multidimensional representation using shearlets. In: Wavelets XI, International Society for Optics and Photonics, p 59140U

[24] Labate D, Safari K, Karantzas N, et al (2019) Structured receptive field networks and applications to hyperspectral image classification. In: Wavelets and Sparsity XVIII, International Society for Optics and Photonics, pp 218–226

[25] Liu G, Reda FA, Shih KJ, et al (2018) Image inpainting for irregular holes using partial convolutions. Proc European Conference on Computer Vision pp 85–100

[26] Liu Y, Pan J, Su Z (2019) Deep blind image inpainting. In: Proc. International Conference on Intelligent Science and Big Data Engineering, pp 128–141

[27] Long J, Shelhamer E, Darrell T (2015) Fully convolutional networks for semantic segmentation. In: Proc. Conference on Computer Vision and Pattern Recognition, pp 3431–3440

[28] Mairal J, Elad M, Sapiro G (2008) Sparse representation for color image restoration. IEEE Transactions on Image Processing 17(1):53–69

[29] Mairal J, Sapiro G, Elad M (2008) Learning multiscale sparse representations for image and video restoration. Multiscale Modeling and Simulation 7(1):214–241

[30] Mallat S (1999) A wavelet tour of signal processing. Elsevier

[31] Pathak D, Krahenbuhl P, Donahue J, et al (2016) Context encoders: Feature learning by inpainting. In: Proc. Conference on Computer Vision and Pattern Recognition, pp 2536–2544

[32] Shen J, Chan TF (2002) Mathematical models for local nontexture inpaintings. SIAM Journal on Applied Mathematics 62(3):1019–1043

[33] Shen L, Xu Y, Zeng X (2016) Wavelet inpainting with the ℓ_0 sparse regularization. Applied and Computational Harmonic Analysis 41(1):26–53

[34] Szegedy C, Liu W, Jia Y, et al (2015) Going deeper with convolutions. In: Proc. Conference on Computer Vision and Pattern Recognition, pp 1–9

[35] Wang Y, Chen YC, Tao X, et al (2020) VCnet: a robust approach to blind image inpainting. In: Proc. European Conference on Computer Vision, pp 752–768

[36] Xie J, Xu L, Chen E (2012) Image denoising and inpainting with deep neural networks. Proc Advances in Neural Information Processing Systems pp 341–349

[37] Yi Z, Tang Q, Azizi S, et al (2020) Contextual residual aggregation for ultra high-resolution image inpainting. In: Proc. Conference on Computer Vision and Pattern Recognition, pp 7508–7517

[38] Yu J, Lin Z, Yang J, et al (2018) Generative image inpainting with contextual attention. In: Proc. Computer Vision and Pattern Recognition, pp 5505–5514

[39] Yu J, Lin Z, Yang J, et al (2019) Free-form image inpainting with gated convolution. In: Proc. International Conference on Computer Vision, pp 4471–4480

[40] Zhang Y, Tiňo P, Leonardis A, et al (2021) A survey on neural network interpretability. IEEE Transactions on Emerging Topics in Computational Intelligence 5(5):726–742

[41] Zhou B, Lapedriza A, Khosla A, et al (2018) Places: A 10 million image database for scene recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 40(6):1452–1464