FUNKNN: NEURAL INTERPOLATION FOR FUNCTIONAL GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Can we build continuous generative models which generalize across scales, can be evaluated at any coordinate, admit calculation of exact derivatives, and are conceptually simple? Existing MLP-based architectures generate worse samples than the grid-based generators with favorable convolutional inductive biases. Models that focus on generating images at different scales do better, but employ complex architectures not designed for continuous evaluation of images and derivatives. We take a signal-processing perspective and treat continuous signal generation as interpolation from samples. Indeed, correctly sampled discrete images contain all information about the low spatial frequencies. The question is then how to extrapolate the spectrum in a data-driven way while meeting the above design criteria. Our answer is FunkNN—a novel convolutional network which learns how to reconstruct continuous images at arbitrary coordinates and can be applied to any image dataset. Combined with a discrete generative model it becomes a functional generator which can act as a prior in continuous ill-posed inverse problems. We show that FunkNN generates high-quality continuous images and exhibits strong out-of-distribution performance thanks to its patch-based design. We further showcase its performance in several stylized inverse problems with exact spatial derivatives.

1 INTRODUCTION

Generative models are used to characterize image classes in applications including solving ill-posed inverse problems Shah & Hegde (2018); Bora et al. (2017), uncertainty quantification Khorashadizadeh et al. (2022) and variational inference Rezende & Mohamed (2015). They learn the desired distributions and produce image samples that are supported on discrete grids. This, however, involves a key limitation—these generative networks are restricted to produce images only at the resolution seen during training. This is impractical for real world applications since images are inherently continuous in nature, and depending upon the physical properties of the signal acquisition device, the obtained images can have a resolution different from what the networks were originally trained on. It is, therefore, highly valuable to have networks that can learn continuous image distributions, thereby making them independent of the image grids of the training distribution.

There has been considerable work on learning grid-free image representations over the past few years. Implicit neural networks (Sitzmann et al., 2020) have been used to produce mesh-free image representations in various inverse problems Chen et al. (2021); Park et al. (2019); Mescheder et al. (2019); Chen & Zhang (2019); Vlašić et al. (2022); Sitzmann et al. (2020). An implicit network regards a given image as a discretized version of a continuous function. The network thereby models it by approximating the image intensity at spatial coordinate $\mathbf{x} \in \mathbb{R}^d$ with a function $f_\theta(\mathbf{x})$, often characterized by a multi-layered perceptron (MLP). While $f_\theta(\mathbf{x})$ can only represent a single image, prior works incorporate a latent code $\mathbf{z}$ in $f_\theta(\mathbf{x}, \mathbf{z})$ to model distributions of continuous images. Here, the implicit network generates a different image from the distribution for each choice of $\mathbf{z}$.

While these prior approaches exhibit good performance in modeling simple datasets, their performance on complex data like human faces is far inferior to that of conventional grid-based generative models based on convolutional neural networks (CNNs) Chen & Zhang (2019); Dupont et al. (2021); Park et al. (2019). This, in fact, holds true even when evaluated over the image resolution they were originally trained on. One of the reasons for their limited performance is that these implicit models

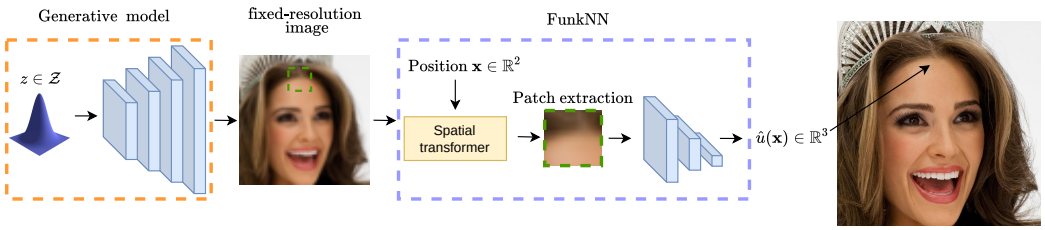


Figure 1: FunkNN architecture. The generative model (orange) produces a fixed-resolution image that is differentiably processed to produce the image intensity at any location (blue).

use MLPs which are not well-suited for modelling image data. In addition, unlike their grid-based counterparts, these implicit generative models suffer from an extra overhead—they require the use of a separate encoding hyper-network Chen & Zhang (2019); Chen et al. (2021); Dupont et al. (2021) or parameter training Park et al. (2019) to obtain latent codes $\mathbf{z}$.

In this paper, we alleviate the above challenges with a new mesh-free convolutional image generator that can faithfully learn the distribution of continuous image functions. The key feature of the proposed framework is our novel patch-based continuous super-resolution network (FunkNN) that takes in a discrete image at any resolution and super-resolves it to generate image intensities at arbitrary spatial coordinates. As shown in Figure 1), our approach combines a traditional fixed size image generator with FunkNN, to result in a deep generative model that can produce images at any arbitrary coordinates or resolution and relies on convolutional layers. Thanks to the flexibility of our approach, we can combine FunkNN with *any* off-the-shelf pre-trained image generator. This includes highly successful GAN architectures Karras et al. (2019; 2020), normalizing flows Kingma & Dhariwal (2018); Kothari et al. (2021) or score-matching generative models Song & Ermon (2019); Song et al. (2020). This, therefore, naturally enables us to learn complex image distributions.

Unlike prior works, FunkNN neither requires a large encoder to generate latent codes nor does it use any MLPs. This is achieved thanks to FunkNN's unique way of integrating spatial coordinate $\mathbf{x}$ with image features. The key idea we exploit is that resolving image intensity at any coordinate $\mathbf{x}$ should only depend on a small local neighborhood around it. Therefore, instead of generating a code for the entire image and then combining it with $\mathbf{x}$ using an MLP, FunkNN simply crops a small patch around $\mathbf{x}$ in the low-resolution image obtained from a traditional generator. This window is then provided to a small convolutional neural network that generates the image intensity at $\mathbf{x}$. The window cropping is performed in a differentiable manner with a spatial transformer network Jaderberg et al. (2015) using bilinear interpolation option.

We experimentally show that FunkNN reliably learns to resolve images to resolutions much higher than those seen during training. In fact, it performs comparably to state-of-the-art continuous super-resolution network Chen et al. (2021) despite having a fraction of the latter's trainable parameters. Unlike traditional learning based methods, our approach can also super-resolve images that belong to image distributions never seen during training. This is because FunkNN only processes small local patches, thus reducing the chance of overfitting on global image features. In addition, we show that our overall generative model framework can produce high quality image samples at any resolution. With the continuous map between spatial coordinates and image intensities provided by our model, we can access spatial image derivatives at arbitrary coordinates and use them to solve a PDE-based inverse problem—an otherwise computationally difficult task with traditional PDE solvers.

2 Implicit neural networks for continuous image representation

Let $u \in L^2(\mathbb{R}^D)^c$ denote a continuous signal of interest with $c \geq 1$ channels and $\mathbf{u}$ be its discretized version supported along a fixed grid with $n \times n \in \mathbb{N}^*$ elements. For simplicity we consider the case with $D = 2$. Our discussion naturally extends to higher dimensions.

An implicit neural network (INN) aims to approximate u by $f_{\boldsymbol{\theta}} : \mathbb{R}^2 \mapsto \mathbb{R}$ parameterized by K weights $\boldsymbol{\theta} \in \mathbb{R}^K$. In standard approaches, the weights are obtained by solving

$$\boldsymbol{\theta}^*(u) = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n^2} \|f_{\boldsymbol{\theta}}(\mathbf{x}_i) - \mathbf{u}(\mathbf{x}_i)\|_2^2, \tag{1}$$

where $\mathbf{u}(\mathbf{x}_i)$ is the image target value sampled at spatial coordinate $\mathbf{x}_i$. This way, implicit neural networks can provide a continuous interpolation for $\mathbf{u}$. Notice, however, that their interpolation scheme is quite arbitrary—solely depending on the choice of network architecture—and does not exploit any image statistics.

From (1), we can observe that the obtained weights $\boldsymbol{\theta}^*(u)$ apply only to a single image $\mathbf{u}$, thereby making $f_\theta(\mathbf{x})$ incapable of providing interpolating representations for a class of images. Some prior works address this by incorporating a low dimensional latent code $\mathbf{z}$ in the implicit network. The new representation then is given by the map $\mathbf{x} \mapsto f_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$, where each $\mathbf{z}$ represents a different image. Note that this framework often entails an additional overhead to relate the images with the latent codes. For example, Chen & Zhang (2019) and Dupont et al. (2021) jointly train f_θ with an encoder network to generate the latent codes for images. Park et al. (2019), on the other hand, optimize the codes as trainable parameters. Figure 2 shows a general illustration of implicit generative networks.

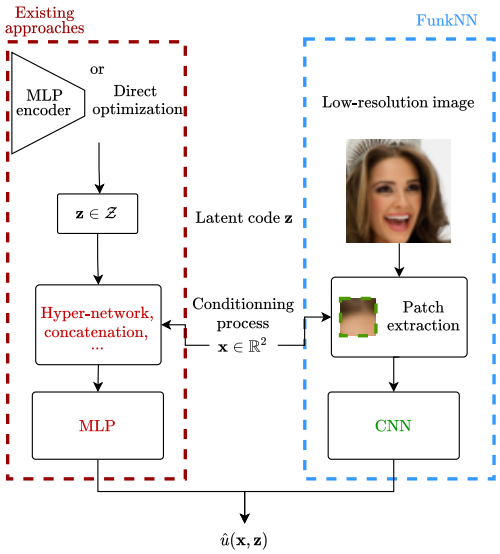


Figure 2: Conceptual difference between FunkNN and existing implicit neural representation.

The prior approaches to model continuous image distributions suffer from a variety of limitations. They often exhibit poor performance in comparison to their grid-based generative counterparts Chen & Zhang (2019); Dupont et al. (2021). This is due to their reliance on MLP networks which do not possess the right inductive biases to model images.

3 Our Approach

We address the challenges discussed in Section 2 by proposing a convolutional image generative model capable of producing continuous images. The proposed framework relies on FunkNN, our novel continuous super-resolution network, capable of interpolating discrete images to arbitrary resolution. Our approach first samples discrete images from a powerful convolutional generator which is followed by interpolation with FunkNN to the continuous domain. Note that we can use any pre-trained discrete generator prior to FunkNN which enables us to learn complex data distributions.

The key advantage of FunkNN over prior works is its architectural simplicity and interpretability from a signal processing perspective. The main idea is that super-resolving a sharp feature from a low resolution image at a spatial coordinate $\mathbf{x}$ should only use a small local neighbourhood around $\mathbf{x}$. This is because high frequency structures like edges and singularities are spatially localized, and only have small range spatial correlations with the low frequency components in the image. FunkNN, therefore, selects a small patch around coordinate $\mathbf{x}$ and passes it to a small CNN to regress image intensity at $\mathbf{x}$. Figure 2 illustrates the differences between FunkNN and prior approaches.

In the subsequent sections, we provide a formal description of FunkNN and our continuous image generative framework.

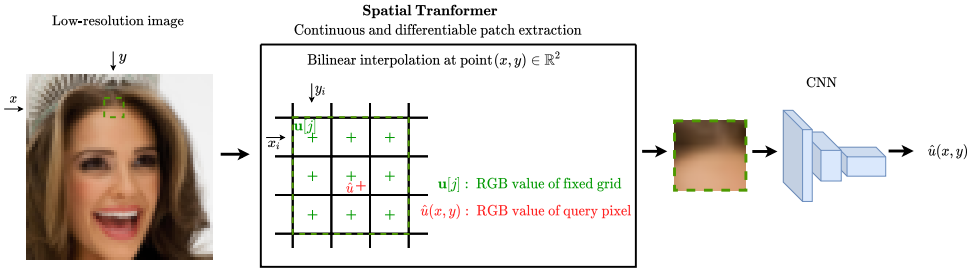


Figure 3: FunkNN architecture. Given a location (x, y), the Spatial Transformer extracts a patch in the discrete image and a CNN produces the intensity at the query position.

3.1 FunkNN: Continuous Super-Resolution Network

Given a low resolution image u, FunkNN estimates the image intensity at a spatial coordinate $\mathbf{x} \in \mathbb{R}^2$ by a two-step process: i) it first extracts a $p \times p$ sized patch from u around the coordinate $\mathbf{x}$, ii) a CNN then maps the patch to the estimated intensity. More formally, if we denote $\mathrm{ST} : (\mathbb{R}^2, \mathbb{R}^{n \times n \times c}) \to \mathbb{R}^{p \times p \times c}$ to be a patch extraction module and $\mathrm{CNN}_{\boldsymbol{\theta}} : \mathbb{R}^{p \times p \times c} \to \mathbb{R}^c$ to be a CNN block that estimates pixel intensity at $\mathbf{x}$ in $\mathbb{R}^c$ from the $p \times p$ patch, then FunkNN can be characterized as

$$\mathrm{FunkNN}_{\boldsymbol{\theta}} \stackrel{\text{def.}}{=} \mathrm{CNN}_{\boldsymbol{\theta}} \circ \mathrm{ST}.$$

Note that patch extraction in the ST module can not be done by trivially cropping it from the image. This is because doing so does not preserve derivatives of image intensities with respect to spatial coordinates—a property useful while solving PDE based inverse problems. We address this by modelling ST with a spatial transformer network Jaderberg et al. (2015). As illustrated in Figure 3, spatial transformers map the low resolution image and input coordinate to the cropped patch in a continuous manner, thereby preserving differentiability. These derivatives can, in fact, be exactly computed using auto-differentiation in PyTorch Paszke et al. (2019) and TensorFlow Abadi et al. (2016) (Figure 4). For further details, please refer to the Appendix A.2.

3.1.1 Training FunkNN

Let $(\mathbf{u}_m)_{1 \leq m \leq M}$ denote a dataset of M discrete images with the maximum resolution $n \times n$. To train the model, we sample a low-resolution image $\mathbf{u}_{\text{lr}} \in \mathbb{R}^{d \times d \times c}$ from a high-resolution image $\mathbf{u} \in \mathbb{R}^{n \times n \times c}$, where $d < n$. When passing through FunkNN, only a small $p \times p$ patch is extracted from the low-resolution image. Therefore, the training dataset could be composed of images of different sizes. To ease notation, we only consider fixed-resolution images. We optimize the weights of the CNN as follows,

$$\underset{\boldsymbol{\theta} \in \mathbb{R}^K}{\arg\min} \sum_{m=1}^{M} \sum_{i=1}^{d^2} \left| \mathrm{FunkNN}_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{u}_{\text{lr},m}) - u_m(\mathbf{x}_i) \right|^2. \tag{2}$$

One advantage of the propose framework is that we can use mini-batches on both images and sampling locations which enables us to train the model using very low memory.

There are multiple possible strategies to train the model. The first and straightforward strategy, which we call *single* mode, is to generate low resolution images in a single size, e.g. $d = n/2$. The other option is *continuous* mode where we generate low-resolution images at different scales, e.g., $d = n/s$ for $s = [s_{\min}, s_{\max}]$ for which the model learns to super-resolve for multiple factors between $s_{\min}$ and $s_{\max}$. The third option is *factor* mode. In this scenario, the model is trained to super-resolve from resolution $r \times r$ to $sr \times sr$ for a single fixed factor s. In this case, once the model is trained, we can generate higher resolution images using a *hierarchical* process, starting from a low resolution image with size $d \times d$, generate images in size $sd \times sd$ and take it as input to the model to generate images with size $s^2 d \times s^2 d$ and go higher.

3.2 GENERATIVE NETWORK

To generate continuous samples, we can combine FunkNN with *any* fixed-resolution discrete convolutional generator. This flexibility allows our overall model the to learn complex continuous image distributions, and to produce images and their spatial derivatives at arbitrary resolutions.

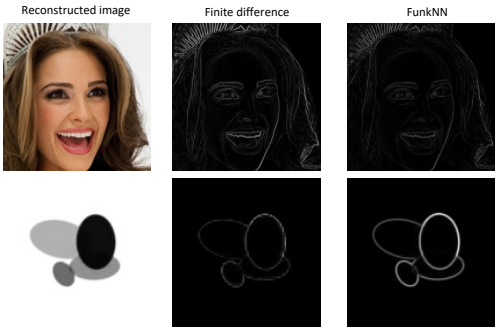


Figure 4: Spatial derivatives with FunkNN.

The choice of fixed-sized generator we use prior to FunkNN depends on the downstream application. While StyleGAN Karras et al. (2019) is arguably the most popular for high quality image generation, GANs exhibit some difficulties when used as generative priors for inverse problems Bora et al. (2017); Kothari et al. (2021). Normalizing flows can alleviate these challenges Whang et al. (2021); Asim et al. (2020); Kothari et al. (2021); Khorashadizadeh et al. (2022), but they require large memory requirements and are expensive to train. While injective encoders offer an effective remedy by using a small network and latent space Kothari et al. (2021), due to architectural limitations posed by injectivity, here we use a regular auto-encoder to represent images in a lower dimensional space, and a low-dimensional flow to match the distribution of latent codes similarly to injective generators. The standard training procedure suggested by Brehmer & Cranmer (2020); Kothari et al. (2021) uses the mean-square error loss which leads to a loss of high-frequency components. We remedy this by using a perceptual loss Johnson et al. (2016). We show in Section 4 that this new idea allows to efficiently solve different ill-posed inverse problems. The network architecture and training details are given in Appendix A.1.

4 CONTINUOUS GENERATIVE MODELS AND SOLVING INVERSE PROBLEMS

In this section, we illustrate the benefits of FunkNN in regularizing and solving ill-posed inverse problems. We focus on the scenario where a generative neural network has been trained on a low-resolution dataset, but we aim at high-resolution reconstruction. This situation is common for many acquisition systems. In experimental science, constructing a good dataset requires painstaking manual supervision and inevitably depends on the particularities of the acquisition system. Due to hardware improvement, the size of the acquisitions will change, rendering earlier priors moot. We illustrate the versatility of the FunkNN architecture to leverage these low-resolution priors. We focus on two problems: solving a simple PDE, inverse problems for a PDE, and limited-view computed tomography (CT).

In the following, we assume that we have low-resolution generative prior on $d \times d$-pixel images, $G : \mathbb{R}^L \to \mathbb{R}^{d \times d \times c}$, which takes as input a latent code $\mathbf{z} \sim \mathcal{N}(0, \mathrm{Id}_L)$. We show that our data driven super-resolution architecture allows interpolating the prior of the generator in a meaningful way to solve complicated problems.

4.1 INVERTING DERIVATIVES

We first apply FunkNN to reconstruct images $u(\boldsymbol{x})$ from samples of spatial derivatives. Related ill-posed reconstructions appear in different domains such as computer vision Park et al. (2019) or obstacle scattering Vlašić et al. (2022). We use the exact spatial derivatives provided by FunkNN and a pre-trained generative model to obtain the latent code z^* that gives us output aligned with the given derivatives,

$$\mathbf{z}^* = \arg\min_{\mathbf{z}} \sum_{j=1}^{n^2} \|\nabla_{\mathbf{x}} \mathrm{IN}_{\boldsymbol{\theta}}(\mathbf{x}_j, G(\mathbf{z})) - \nabla_{\mathbf{x}} u(\mathbf{x}_j)\|_2^2 + \lambda \|\mathbf{z}\|_2^2. \tag{3}$$

We also found that updating the weights $\boldsymbol{\theta}$ of the generative model further helps in producing better reconstruction. The exact procedure is detailed in Appendix B.1. The estimated reconstruction is given by sampling the trained FunkNN network on the high-resolution grid: $\hat{u}_j = \mathrm{IN}_{\boldsymbol{\theta}}(\mathbf{x}_j, G(\mathbf{z}^*))$.

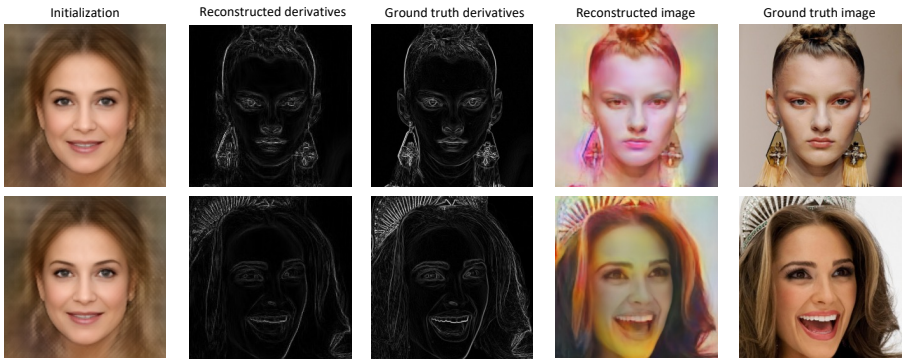


Figure 5: Reconstructing an image from its derivatives.

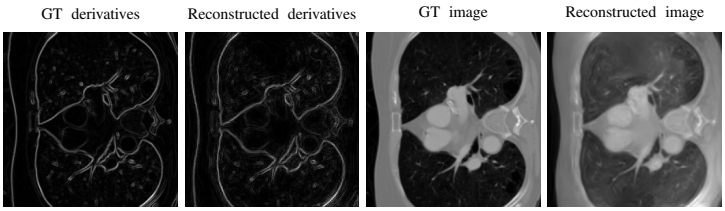


Figure 6: Solving PDE inverse problem with missing derivatives.

An important consequence of how we designed FunkNN is that it allows us to compute exact spatial derivatives via automatic differentiation. This is important for solving PDE-based forward and inverse problems with fields or unknowns parameterized by FunkNN. Figure 5 illustrates FunkNN's performance, which provides high-quality spatial reconstructions from the observation of the spatial derivatives. Notice that Problem (3) is ill-posed and it is not possible to accurately estimate mean intensity of the original image only based on the derivatives. Despite the different color scales, the unknown image is accurately estimated by FunkNN.

4.2 Sparse derivatives

We now make the previous problem even more ill-posed by observing not the spatial derivatives on a dense grid, but only 20% of them that have the highest intensity. It amounts to only sample the gradient on the $n_s < n^2$ locations that gives the largest value $\|\nabla_{\mathbf{x}} u(\mathbf{x}_j)\|$. In order to account for the missing low-amplitude gradient, we add an extra total-variation regularization term. Thus, we aim at solving

$$\mathbf{z}^* = \arg\min_{\mathbf{z}} \sum_{j=1}^{n_s} \|\nabla_{\mathbf{x}} \mathrm{IN}_{\boldsymbol{\theta}}(\mathbf{x}_j, G(\mathbf{z})) - \nabla_{\mathbf{x}} u(\mathbf{x}_j)\|_2^2 + \lambda \|\mathbf{z}\|_2^2 + \lambda_2 \|\nabla G(\mathbf{z})\|_2, \tag{4}$$

using the same process as in the previous setting (see Appendix B.1).

The reconstruction results are shown in Figure 6. First row contains the true set of derivatives (left), where only the 20% with the highest intensity are observed, and the estimated derivatives (right). Second row is the true image (left) and the reconstruction (right). This example of highly under-determined system shows that the proposed FunkNN architecture can leverage existing prior in addition to super-resolving both an image and its derivative. Moreover, the estimated spatial derivatives are meaningful even when the prior only provide the grid-based images.

4.3 Limited-view CT

In CT, we observe projections of a 2-dimensional image $u \in L^2(\mathbb{R}^2)$ through the (discrete angle) Radon operator $\mathcal{A}(\boldsymbol{\alpha}) : L^2(\mathbb{R}^2) \to L^2(\mathbb{R})^I$ where $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_I\}$ denotes the I viewing direc-

tions with $\alpha_i \in [0, 2\pi)$. Finally, the discrete observation is given by

$$\mathbf{v}_i = (\mathcal{A}(\alpha_i)u)(\mathbf{x}) + \boldsymbol{\eta}_i, \tag{5}$$

where $\mathbf{x} = [\mathbf{x}_1, \ldots, \mathbf{x}_{n^2}]$ is the $n \times n$ sampling grid and $\boldsymbol{\eta}_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \mathrm{Id}_{n^2})$ is a random perturbation.

If the view directions correspond to a fine sampling of the interval $[0, 2\pi)$, then the adjoint operator $\mathcal{A}^*$, namely the filtered back-projection (FBP) operator, allows to recover the original signal even in the presence of noise. For application-specific reasons it is not always possible to observe the full viewing direction set, leading to limited-view CT. Here we sample (α_i) uniformly between -70 degrees and $+70$ degrees. This incomplete set of observations makes the estimation of the signal u from (5) strongly ill-posed. We address limited-view CT using the FunkNN framework:

$$\mathbf{z}^* = \underset{\mathbf{z} \in \mathbb{R}^L}{\arg\min} \sum_{i=1}^{I} \|\mathcal{A}(\alpha_i)\mathrm{IN}_{\boldsymbol{\theta}}(\mathbf{x}, G(\mathbf{z})) - \mathbf{v}_i\|_2^2 + \lambda \|\mathbf{z}\|_2^2, \tag{6}$$

In Figure 7, we display the results obtained for images from a different dataset. The precise experimental setting is relegated to Appendix C. Notice that despite the noise in the projections and the incomplete set of viewing directions, leveraging the learned low-resolution prior in solving Problem (5) provides an informative estimation. This spectacular reconstruction is mainly due to the strong prior brought in by the generative model. Then our FunkNN architecture allows us to use this prior in a higher resolution. Even though very fine details are not necessarily recovered by the reconstruction, the proposed approach provides a way to perform reconstruction in this very ill-posed setting, e.g. see the output of FBP algorithm. The gains brought in by the super-resolution block are assessed individually in Section 5. We emphasize that to the best of our knowledge, this is a novel way of solving this problem.

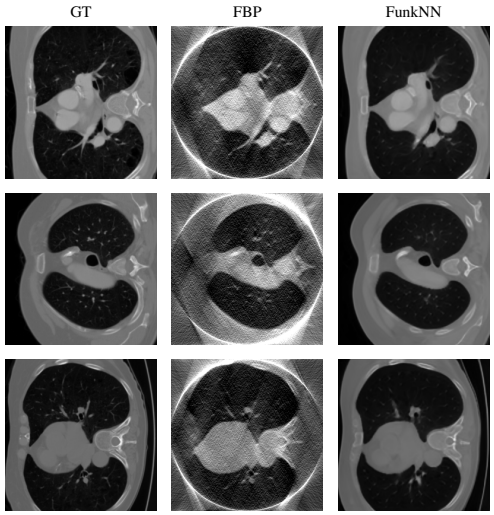


Figure 7: Limited-view CT reconstruction.

5 Experimental Results

5.1 Super Resolution

We first showcase the continuous, arbitrary-scale super-resolution by FunkNN. We work on the CelebA-HQ Karras et al. (2017) dataset with resolution 128×128, and we train the model using three training strategies explained in section 3.1.1, *single*, *continuous* and *factor*.

The model *never sees training data at resolutions 256 × 256 or higher*. Figure 8 shows the performance of FunkNN trained using *factor* strategy, which indicates that FunkNN yields high-quality image reconstructions at much higher resolutions than it was trained on. This experiment shows that FunkNN learns an accurate super-resolution operator and generalizes to resolutions not seen during training. We perform successive super-resolution up-scaling using the proposed network, increasing the size of the original image by a factor of 2 until reaching a resolution of 1024×1024. Our model demonstrates comparable performance to the state-of-the-art baseline LIIF-EDSR Chen et al. (2021) but with a considerably smaller network; our model only has 190K trainable parameters while LIIF uses 1.5M parameters. Quantitative results are provided in Table 1.

5.2 Robustness to Out-of-Distribution Samples

One exciting aspect of the proposed model is its strong generalization to out-of-distribution data– a boon obtained from FunkNN's patch based operation. We showcase it by evaluating FunkNN originally trained on CelebA-HQ Karras et al. (2017) at resolution 128×128 (Section 5.1), to super-resolve images from the LSUN-bedroom Yu et al. (2015) dataset which are structurally very different

Figure 8: FunkNN model is only trained over images of maximum resolution 128×128, but it can reliably reconstruct high-quality images in higher resolutions.

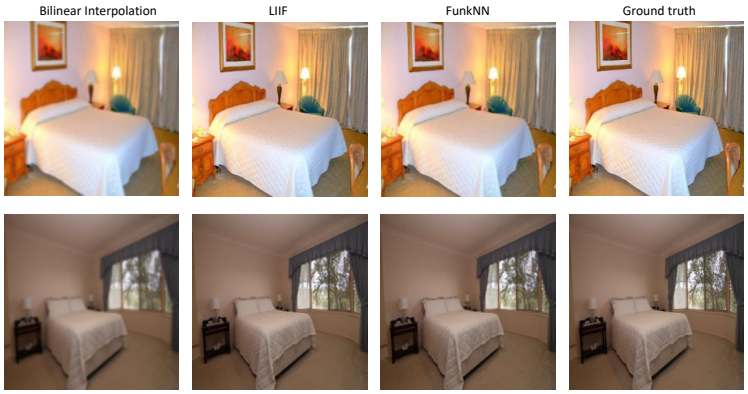


Figure 9: Image super-resolution on 256×256 LSUN images while the model is trained on 128×128 images from CelebA-HQ data.

from the training data. Moreover, we super-resolve images from size 128 to 256, which were not seen at training time. Figure 9 indicates that the proposed model can faithfully super-resolve out-of-distribution images at new resolutions.

Table 1: Performance of different methods over super-resolution in SNR(dB)

	$128 \to 256$	$128 \to 512$	$128 \to 1024$	$128 \to 256$ (OOD)
Bilinear Interpolation	28.8	26.4	24.1	23.6
LIIF-EDSR Chen et al. (2021)	**31.2**	27.6	23.5	26.0
FunkNN (single)	**31.2**	27.0	23.9	**26.2**
FunkNN (continuous)	30.0	**27.7**	24.5	25.7
FunkNN (factor)	**31.2**	27.6	**24.6**	25.9

6 Related Works

6.1 Generative models

While multilayer perceptrons (MLPs) have shown promising results to represent a single object Tancik et al. (2020); Sitzmann et al. (2020); Martel et al. (2021); Saragadam et al. (2022), they usually lack expressivity to generate real world data like human faces Chen & Zhang (2019); Dupont et al.

(2021). On these complex datasets, the quality of the generation by MLP is still behind convolutional image generators Radford et al. (2015); Karras et al. (2019; 2020). The superiority of CNN has been established in various other applications of image processing, see Krizhevsky et al. (2017) for a benchmark on different tasks. One reason often stated is a stronger inductive bias for natural images of CNN over MLP networks.

As a step toward generating set of continuous signals, the authors of Ntavelis et al. (2022) propose to network that returns a grid around a given position, which makes the GAN shift and scale invariant. The idea of combining GAN with implicit neural network as already been proposed, especially for rendering of 3D shapes where it shows impressive results Chan et al. (2021); Schwarz et al. (2020). First approaches of trully continuous output work on very simple binary shapes Chen & Zhang (2019); Park et al. (2019); Mescheder et al. (2019). Then Anokhin et al. (2021); Xu et al. (2021) investigate the reason of the success of implicit representation in GAN. More recently, Dupont et al. (2021) proposed a more expressive model to learn the distribution of functions based on adversarial training and Fourier features Tancik et al. (2020) followed by an MLP network.

Another popular idea to represent a set of signals is to use hypernetwork that parameterized another network that are known to work well for a single image. This natural idea can be found in various fields to capture set of discrete images Klocek et al. (2019), continuous images Skorokhodov et al. (2021) or for scene renderingSitzmann et al. (2019). More complex architectures can also allow network with carefully designed layers Dupont et al. (2021) or with strong properties such as injectivity Alberti et al. (2022).

As explain in the first section, having a GAN with a continuous output doesn't necessarily mean that we have access to a high-resolution signal. The idea of having a GAN that generate super-resolved output is only very recent. In Chai et al. (2022), the authors introduce a GAN architecture that produces continuous image with a possibility to 'zoom in' the output signal, while still generating new details. However, this architecture can be difficult to train and doesn't benefit from all the properties of implicit representation, such as having access to exact spatial derivative.

6.2 Super-resolution

Training an implicit representation to describe a discrete signal is a tempting way to define an infinite resolution representation of the given signal. Unfortunately, the way implicit neural network are trained doesn't allow to learn fine details between sampling points, usually because they are not known even at training. Retrieving these lost details is the field of super-resolution.

Recent works incorporate machine learning to adapt the well established theory of interpolation. The tempting connection between these two fields is formalized by learning the weights of a convolution kernel Jia et al. (2017); Hu et al. (2019) or by learning to aggregate several interpolation methods Dong et al. (2015); Kim et al. (2016); Pandey & Ramakrishnan (2018). Each approaches focus on a careful training procedure to ensure learning to super-resolved since the standard interpolation methods is not designed for super-resolution.

7 Conclusion

Despite images being inherently continuous signals, traditional generative models characterize them only on discrete grids. Recently proposed networks generate continuous samples by relying on fully connected layers and thereby lack the right inductive biases to model images effectively. In this paper, we alleviate these challenges in two steps. (i) Instead of generating continuous images directly, we first use a convolutional generator pre-trained to sample discrete images from the desired distribution. (ii) We then use FunkNN, our novel continuous super-resolution architecture, to resolve the discrete images produced by the generator to arbitrary resolution. Our experiments show that we obtain performance comparable to state-of-the-art networks for continuous super-resolution despite using only a fraction of trainable parameters. Our generative model also reliably solves various stylized first order PDE based inverse problems. In addition, our proposed approach is highly flexible and can directly be used with *any* off-the-shelf generative network.

REFERENCES

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pp. 265–283, 2016.

Giovanni S Alberti, Matteo Santacesaria, and Silvia Sciutto. Continuous generative neural networks. *arXiv preprint arXiv:2205.14627*, 2022.

Ivan Anokhin, Kirill Demochkin, Taras Khakhulin, Gleb Sterkin, Victor Lempitsky, and Denis Korzhenkov. Image generators with conditionally-independent pixel synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14278–14287, 2021.

Muhammad Asim, Max Daniels, Oscar Leong, Ali Ahmed, and Paul Hand. Invertible generative models for inverse problems: mitigating representation error and dataset bias. In *International Conference on Machine Learning*, pp. 399–409. PMLR, 2020.

Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G Dimakis. Compressed sensing using generative models. In *International Conference on Machine Learning*, pp. 537–546. PMLR, 2017.

Johann Brehmer and Kyle Cranmer. Flows for simultaneous manifold learning and density estimation. *Advances in Neural Information Processing Systems*, 33:442–453, 2020.

Lucy Chai, Michael Gharbi, Eli Shechtman, Phillip Isola, and Richard Zhang. Any-resolution training for high-resolution image synthesis. *arXiv preprint arXiv:2204.07156*, 2022.

Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5799–5809, 2021.

Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8628–8638, 2021.

Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5939–5948, 2019.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2): 295–307, 2015.

Emilien Dupont, Yee Whye Teh, and Arnaud Doucet. Generative models as distributions of functions. *arXiv preprint arXiv:2102.04776*, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Xuecai Hu, Haoyuan Mu, Xiangyu Zhang, Zilei Wang, Tieniu Tan, and Jian Sun. Meta-sr: A magnification-arbitrary network for super-resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1575–1584, 2019.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28, 2015.

Xu Jia, Hong Chang, and Tinne Tuytelaars. Super-resolution with deep adaptive image resampling. *arXiv preprint arXiv:1712.06463*, 2017.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pp. 694–711. Springer, 2016.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8110–8119, 2020.

AmirEhsan Khorashadizadeh, Konik Kothari, Leonardo Salsi, Ali Aghababaei Harandi, Maarten de Hoop, and Ivan Dokmani'c. Conditional injective flows for bayesian imaging. *arXiv preprint arXiv:2204.07664*, 2022.

Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1646–1654, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

Sylwester Klocek, Łukasz Maziarka, Maciej Wołczyk, Jacek Tabor, Jakub Nowak, and Marek Śmieja. Hypernetwork functional image representation. In *International Conference on Artificial Neural Networks*, pp. 496–510. Springer, 2019.

Konik Kothari, AmirEhsan Khorashadizadeh, Maarten de Hoop, and Ivan Dokmanić. Trumpets: Injective flows for inference and inverse problems. In *Uncertainty in Artificial Intelligence*, pp. 1269–1278. PMLR, 2021.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

Johannes Leuschner, Maximilian Schmidt, Daniel Otero Baguer, and Peter Maass. Lodopab-ct, a benchmark dataset for low-dose computed tomography reconstruction. *Scientific Data*, 8(1):1–12, 2021.

Julien NP Martel, David B Lindell, Connor Z Lin, Eric R Chan, Marco Monteiro, and Gordon Wetzstein. Acorn: Adaptive coordinate networks for neural scene representation. *arXiv preprint arXiv:2105.02788*, 2021.

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4460–4470, 2019.

Evangelos Ntavelis, Mohamad Shahbazi, Iason Kastanis, Radu Timofte, Martin Danelljan, and Luc Van Gool. Arbitrary-scale image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11533–11542, 2022.

Ram Krishna Pandey and AG Ramakrishnan. A hybrid approach of interpolations and cnn to obtain super-resolution. *arXiv preprint arXiv:1805.09400*, 2018.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 165–174, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pp. 1530–1538. PMLR, 2015.

Vishwanath Saragadam, Jasper Tan, Guha Balakrishnan, Richard G Baraniuk, and Ashok Veeraraghavan. Miner: Multiscale implicit neural representations. *arXiv preprint arXiv:2202.03532*, 2022.

Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. *Advances in Neural Information Processing Systems*, 33:20154–20166, 2020.

Viraj Shah and Chinmay Hegde. Solving linear inverse problems using gan priors: An algorithm with provable guarantees. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 4609–4613. IEEE, 2018.

Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *Advances in Neural Information Processing Systems*, 32, 2019.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020.

Ivan Skorokhodov, Savva Ignatyev, and Mohamed Elhoseiny. Adversarial generation of continuous images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10753–10764, 2021.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020.

Tin Vlašić, Hieu Nguyen, and Ivan Dokmanić. Implicit neural representation for mesh-free inverse obstacle scattering. *arXiv preprint arXiv:2206.02027*, 2022.

Jay Whang, Qi Lei, and Alex Dimakis. Solving inverse problems with a flow-based noise model. In *International Conference on Machine Learning*, pp. 11146–11157. PMLR, 2021.

Rui Xu, Xintao Wang, Kai Chen, Bolei Zhou, and Chen Change Loy. Positional encoding as spatial inductive bias in gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13569–13578, 2021.

Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

A Network Architectures

A.1 Generative model architecture

The generative networks used to address inverse problems described in Section 4 are trained in two-steps: i) we train an Auto-Encoder (AE) neural network to map the training samples to a lower dimensional latent code, ii) we train a normalizing flow model to learn the distribution of the computed latent codes by using maximum likelihood (ML) loss function.

(a) Ground truth test samples (b) AE reconstructions

Figure 10: AE performance on CelebA-HQ in resolution 128×128.

A.1.1 AUTO-ENCODER ARCHITECTURE

The AE network is a succession of an encoder and a decoder neural networks. The encoder maps images to a low dimensional latent code, a L-dimensional vector. It is a succession of 6 Conv+ReLu blocks (one convolution layer followed by a ReLu activation unit). Between each Conv+ReLu blocks, there is a down-sampling operation. The decoder network transforms the computed latent code to the image samples. The decoder has the same structure as the encoder network, except that the down-sampling operations are replaced by up-sampling layers. In order to increase the expressivity of the model, we equip our model with local skip connections in both encoder and decoder, as suggested in He et al. (2016). The latent code dimension is 512 for training data in resolution 128×128 for both RGB and gray-scale images. The AE model has around 40M and 11M trainable parameters for limited-view CT and CelebA-HQ experiments.

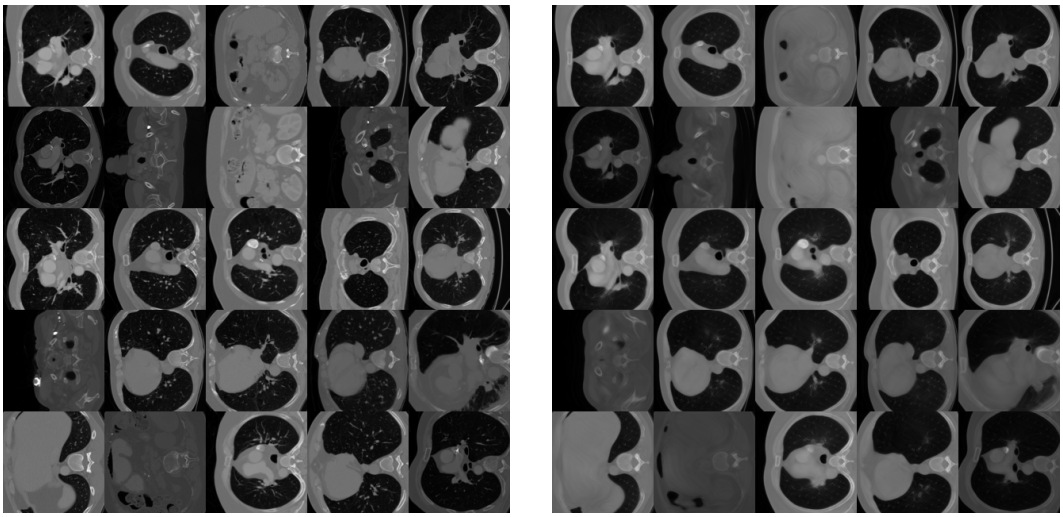
Training strategy While we can train the AE model using only the MSE loss Brehmer & Cranmer (2020); Kothari et al. (2021), it often suppresses the high-frequency components of the reconstructed image. To alleviate this issue, we combine the perceptual loss proposed in Johnson et al. (2016) with regular MSE loss which significantly improves the quality of the reconstructions.

Experimental Result: We train AE model over 40000 images of the LoDoPaB-CT Leuschner et al. (2021) and 30000 images of CelebA-HQ Karras et al. (2017) datasets in resolution 128×128. We train the model over CT and CelebA-HQ images for 600 and 200 iterations. Figures 11 and 10 display the output image produce by the AE on the test dataset. Notice that very small details are lost by the AE. However, the final reconstruction contains the most important details. We show that this is enough to learn an informative prior and solve ill-posed inverse problems. We also show that it is possible to go beyond the quality of reconstruction given by the trained AE, see Section 4.3.

A.1.2 NORMALIZING FLOW

We use RealNVP Dinh et al. (2016) architecture with fully-connected layers for scale and bias subnetworks with two hidden layers of dimension 1024 and 512. The model comprises five masked affine coupling blocks and activation normalization. As mentioned earlier, we train the model using ML loss. The flow model has 10M trainable parameters.

Experimental Results: As soon as the flow model is trained, we take samples from the Gaussian distribution and feed to the flow model to generate new latent codes. Decoder then takes the generated latent codes and produce new images. Figure 12 demonstrates the generated samples by our generative model (flows + AE) for CT and CelebA-HQ dataset.



(a) Ground truth test samples (b) AE reconstructions

Figure 11: AE performance on LoDoPaB-CT in resolution 128×128.

(a) CT (b) CelebA-HQ

Figure 12: Generative model performance on CT and CelebA-HQ samples in resolution 128×128.

A.2 FUNKNN ARCHITECTURE

We crop a patch with size $p = 8$ from the low-resolution image by using spatial transformer with *bilinear* interpolation kernel and *reflection* mode for the border pixels. We use four convolutional layers with filter size 3 and channel dimension 128 and ReLu activation functions. We use two maxpooling layers with size (2,2) inside to reduce the feature size. We then use two fully-connected layers with latent dimension 100. We also use mini-batches with 512 pixels and 64 objects for each training iteration.

All the models are trained using Adam optimizer Kingma & Ba (2014) with learning rate 10^{-4}.

B INVERSE PROBLEMS: EXPERIMENTAL SETTINGS

B.1 PDE INVERSE PROBLEM: EXPERIMENTAL SETTING

The training strategy of the networks is detailed in Appendix A for the two dataset presented in Section 4.1. The low-resolution images are composed of $d \times d = 128 \times 128$ pixels and the high-resolutions target images are of size $n \times n = 256 \times 256$. The number of sampling points in the setting described in Section 4.2 is $n_s = 13,000$. The estimated solution is obtained by running first 1000 iterations of stochastic gradient descent using Adam optimization algorithm on Problem (3) with $\lambda = 0$ and on Problem (4) with $\lambda = 0$ and $\lambda_2 = 10^{-2}$. Then, we run 1000 iterations of stochastic gradient descent to optimize the weights of the FunkNN network

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{j=1}^{n^2} \|\nabla_{\mathbf{x}} \mathrm{IN}_{\boldsymbol{\theta}}(\mathbf{x}_j, G(\mathbf{z}^*)) - \nabla_{\mathbf{x}} u(\mathbf{x}_j)\|_2^2. \tag{7}$$

This custom training strategy allows a significant gain in the quality of the reconstructions.

The observations given by equation (5) are degraded by additive Gaussian noise such that the SNR of each projection is 30dB.

C LIMITED-VIEW CT: EXPERIMENTAL SETTING

The training strategy of the networks is detailed in Appendix A. The low-resolution images are composed of $d \times d = 128 \times 128$ pixels and the high-resolutions target images are of size $n \times n = 256 \times 256$. The estimated solution is obtained by running 5000 iterations of stochastic gradient descent using Adam optimization algorithm on Problem (5) with $\lambda = 0$. The observations given by equation (5) are degraded by additive Gaussian noise such that the SNR of each projection is 30dB.