Filtering In Neural Implicit Functions

Yixin Zhuang

Peking University

Abstract. Neural implicit functions are highly effective for data representation, including images and 3D surfaces. However, the implicit functions learned by neural networks usually include unexpected artifacts or lose fine details if the input data has many scales of details or a wide range of frequencies. It is challenging to remove artifacts while recovering details from the reconstructed functions and usually comes out with over smoothing or noisy issues. To overcome this, we propose a new framework, coined FINN, that integrates a filtering module into the neural network to perform data reconstruction while filtering unexpected artifacts. The filtering module has a smoothing operator acting on intermediate results of the network and a recovering operator bringing details from the input back to regions overly smoothed. The filtering module enables recovering both low-frequencies and high-frequencies with high fidelity. We demonstrate the advantage of FINN on tasks of image regression and surface reconstruction and showcases significant improvement compared to state-of-the-art methods. In addition, FINN also yields better performance in both convergence speed and network stability.
...

Keywords: Implicit Functions, Filters, Neural Implicit Representations

1 Introduction

Filtering techniques are widely applied to many applications to enhance the content of interest or remove unexpected artifacts from the data, such as images and 3D shapes. For the data with many scales of details or a wide range of frequencies, it is challenging to recover the data while maintaining free of unwanted content using neural networks. Therefore, filtering methods are usually necessary for post-processing, or additional strategies, e.g., coarse to fine generation, hierarchical representations, or scale-aware operations, are needed to adapt different scales of details. This paper addresses built-in learnable filters in a fully connected neural network to achieve both reconstruction and filtering purposes for neural implicit functions within a simple network framework.

An implicit function is a continuous function that maps spatial coordinates to the corresponding values under varying representations (e.g., RGB, signed distance, etc.). It can be learned using simple network structures, i.e., Multi-Layer Perceptrons (MLPs) with ReLU activations [20, 22, 8]. Moreover, MLPs can reconstruct functions with wide frequency bandwidths by encoding the coordinates to high-dimensional vectors, i.e., random Fourier features (FFN) [27],

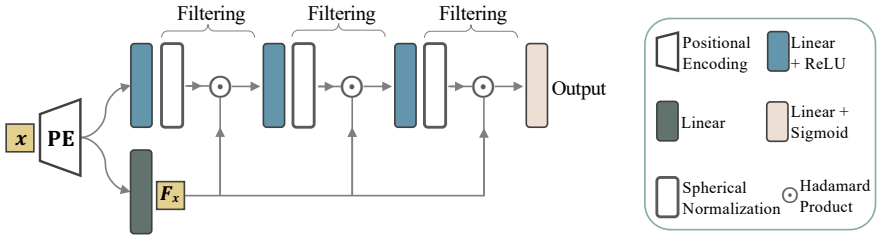
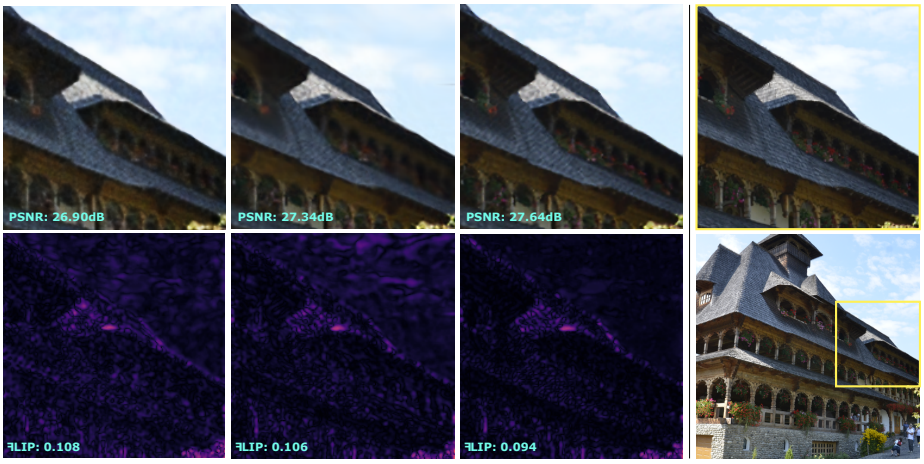


Fig. 1. We present a new framework for learning implicit functions with filters. Filtering artifacts is usually at the expense of losing details. Aiming to achieve adaptive filtering, we incorporate filtering modules into multilayer perceptrons (MLPs) to perform smoothing and recovering operators iteratively at each layer of MLPs. The smoothing operator outputs over-smoothed results, while the recovering operator restores fine details. The filtering on intermediate results of MLPs consequently yields an adaptive reconstruction of the target function achieving high fidelity of neural representations.

or replacing ReLU activation functions with periodic functions, i.e., sinusoidal activation (SIREN) [25], leading to significant improvements.

Despite the impressive progress, FFN and SIREN still have difficulty in reconstructing complex functions and usually create over-smoothed regions or noise in the results. Unfortunately, there is no ubiquitously valid filter to adapt the reconstruction to avoid these issues, and each data needs to be individually adjusted. To this end, some explicit spatial or local adapting schemes are presented. For example, based on FFN, SAPE [13] progressively encodes the input with increasing frequencies for individual spatial coordinates to avoid using excessive frequencies at smooth regions, and based on SIREN, the work [19] subdivides the input coordinates into grids and modulates activation functions according to individual grids. An example is shown in Figure 2, where the explicit spatial adaptation method (i.e., SAPE) provides results with less noisy artifacts compared to the baseline method (i.e., FFN). Nevertheless, SAPE tends to over-smooth the result (e.g., the sky and the regions under roofs). Unlike those adaptation methods, this paper opts for continuous filtering functions to achieve adaptation without discretizing or subdividing the coordinate space and circumvents the overly smoothing issues observed in them.

We define the filtering module as two consecutive operators, a smoothing operator, i.e., spherical normalization, overly smoothes the results, and a recovering operator, i.e., Hadamard product, brings fine details from the input back to the results over smoothed, as shown in Figure 1 (Filtering). The two operators together with the MLPs form a simple network function. Intuitively, the spherical normalization operator flattens both global and small-scale variations of the results. And in contrast, the recovering operator enables small structures or patterns to be brought back from an original domain to the over-smoothed results. With the filtering function, the intermediate results are adjusted layer-wisely from MLPs, and finally, yielding the filtering of the reconstructed function.



(a) FFN (b) SAPE (c) FINN (d) GT

Fig. 2. (a) An image generated by Fourier Features Network (FFN [27]) contains noises (Top) that are highlighted in the error map computed by FLIP [1] (Bottom). (b) SAPE [13] smoothes out noises by spatial adaptation of frequencies, but at the expense of losing small structures and textures, e.g., clusters of clouds and roofs. In contrast, (c) our method (FINN) incorporates a filtering module that effectively removes artifacts while recovering distinct details. PSNR (larger is better) and FLIP (smaller is better) measures similarity between generated results and ground truth images.

Compared to FFN and SAPE in Figure 2, our method showcases improvement with less noisy artifacts and enhanced details.

To demonstrate the effectiveness of the filtering module, we evaluate our network on both image regression and surface reconstruction tasks. Experimental results showcase that our method can capture a wide range of frequencies better than state-of-the-art methods, including FFN and SIREN, and achieves significant performance improvement. Moreover, filtering in neural network functions yields a faster convergence speed and more stable performance under SGD-based optimization.

In this work, we make the following contributions.

- We propose a simple framework that extends MLP-based neural network with filtering functions that perform data reconstruction while filtering noisy artifacts and enhancing small-scale details.
- We devise a simple filtering function with iterative smoothing and recovering operators to each layer of MLPs that can well adapt the reconstruction with a wide range of frequency bandwidths.

In addition, the whole network has a simple formulation that can be easily modified to realize different filtering effects, e.g., using an alternative function for recovering operator, or extended for various applications.

2 Related Work

Implicit Neural Representation. Deep neural network has been shown to be highly effective for learning implicit function that represents images [27, 25, 21, 6] and 3D objects and scenes [22, 20, 8, 2, 12, 26]. They incorporate coordinates as inputs to Multi-Layer Perceptrons (MLPs) that could be sampled to an arbitrarily high spatial resolution. Thus such representation can directly be applied to super-resolution tasks. Other applications include view-synthesis [21, 18, 34, 3], point cloud based 3D reconstruction [2, 12, 30, 29] and single-image 3D reconstruction [20, 31, 32, 24]. Besides generative tasks, implicit representation is also applicable and effective for other tasks, such as feature matching [7] and scene understanding [36].

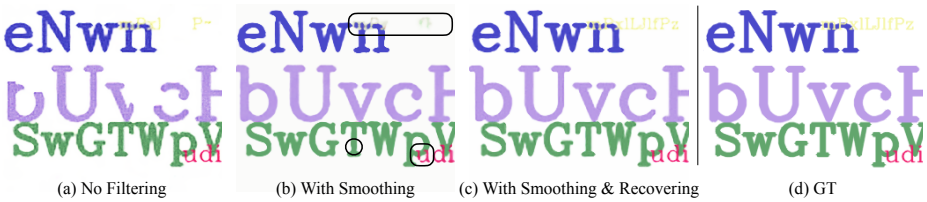
Some recent advances have led to structured or hierarchical designs that can further close the gap between generated results and the target function. They divide the complex functions of 3D object and scenes [4, 10, 15] or image [19] into regular subregions, and fitting each subregions individually while considering globally consistency.

As evidenced by Fourier Feature Networks (FFN) [27], ReLU-based MLP networks are struggling to capture highly detailed signals, due to the spectral bias trait. FFN uses positional encoding to map input coordinates of signals to a high dimensional space using sinusoidal functions. Meanwhile, SIREN [25] replaces the ReLU activations in the MLP network with sinusoidal functions. They share a similar spirit that manipulates the input or intermediate results in the frequency domain to capture a wider frequency bandwidths in the output. Further, to better fit individual data, the sinusoidal functions in positional encoding or MLP networks are devised to be learnable [9, 19]. In contrast to using sinusoidal functions, Spline Positional Encoding (SPE) [28] studies learnable spline functions for coordinate embedding. With sufficient local supports of splines, the SPE can also approximate the signal with high frequencies. However, when using a small number of local supports, the boundaries between patches of the spline become notable that significantly reduce the visual quality.

Reconstructing high-frequency details is commonly at the expense of introducing visual artifacts to the results. Structured or hierarchical designs can hardly take effect if a subregion itself is complex. To this end, SAPE [13] presents a progressive optimization strategy to encode signals with increasing frequencies at individual spatial locations. The method significantly reduces the noisy artifacts, but on the other hand, generates over-smoothed results.

In contrast to the approaches that manipulate the input for spatial adaptation or structured representation, we seek an alternative scheme that directly adapts the continuous functions.

Image Filtering with Deep Neural Networks. Image filters usually compute the weighted average of neighboring pixels of the image as output or sometimes leverage regularization terms for image optimization. Recently, a line of research presents filters with neural networks that can be learned from a large



(a) No Filtering (b) With Smoothing (c) With Smoothing & Recovering (d) GT

Fig. 3. Reconstruction using different combinations of filtering operations (a-c). (a) is generated by FFN [27] with no filtering. With smoothing operation (b), our method generates much complete and smoother texts than FFN. By further applying the recovering operation (c), small structures are recovered based on (b), e.g., the details highlighted in black circles in (b). Under the same sampling rate, i.e., 25% for training and 100% for testing, incorporating a filtering module can significantly improve the visual quality of reconstructed images.

number of datasets for various applications [14, 35, 33, 11, 5, 16], such as image super-resolution, denoising, and deblurring, etc. They are mainly based on the convolutional network from which the local neighborhood of the query point is determined. For continuous functions, it is difficult to be aware of the neighbors except to discretize the input domain with limited resolutions.

Thus, it is still an open problem to enable filtering of the continuous functions in a fully connected neural network. And the purpose of this work is a simple and effective filtering scheme for neural implicit functions.

3 Method

In this section, we introduce a new fully-connected neural network with filtering modules, as illustrated in Figure 1. We first present the neural implicit functions and the positional encoding in Section 3.1. Next, we describe the the filtering module in Section 3.2, and filtering effects in Section 3.3.

3.1 Neural Implicit Functions

An implicit function is a continuous function $f_\theta : \mathbb{R}^a \to \mathbb{R}^n$ which takes as input a coordinate of any query point from the Euclidean space $x \in \mathbb{R}^a$ and predicts a value in the target domain $\mathbb{R}^n$. Learning f_θ with a neural network requires a set of coordinate samplings as input and corresponding values as output. Examples of the f_θ include mapping pixels to intensity values for image regression or projecting 3D coordinates to scalar values for 3D surface reconstruction. Once optimized, the target f_θ can be sampled to an arbitrarily high spatial resolution, which implies the network has generalizability to unseen spatial positions beside the training samples.

It is common to model implicit function with Multi-Layer Perceptrons (MLPs). However, as evidenced by FFN [27], ReLU-based MLPs can hardly fit high-frequency signals due to the spectral bias trait, leading to the severe under-fitting

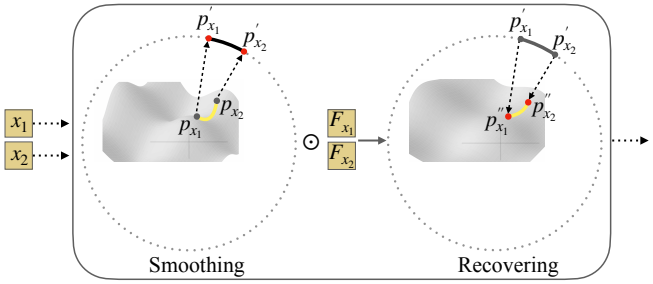


Fig. 4. Illustration of smoothing and recovering operations. Input coordinates x_1, x_2 are projected to features P_{x_1}, P_{x_2} after positional encoding and a fully connected layer. After smoothing (i.e. spherical normalization), local variations between them are smoothed out, as shown by the line segment between $P^{'}_{x_1}, P^{'}_{x_2}$ on the hypersphere. To multiple with initial features F_{x_1}, F_{x_2}, the recovering operation brings $P^{'}_{x_1}, P^{'}_{x_2}$ from the hypersphere back to a more complex manifold, from which the interval between the new features $P^{''}_{x_1}, P^{''}_{x_2}$ have less variations than the original features P_{x_1}, P_{x_2} if F_{x_1}, F_{x_2} are chose properly. This process is iterative performed after each fully connected layers.

problem. Hence, to avoid the bias of favoring the low-frequency spectrum, FFN projects the input to the frequency domain with a controllable frequency bandwidth. Specifically, the coordinate x is represented by a d-dimensional Fourier feature vector, $\gamma(x) \in \mathbb{R}^d$, as follows:

$$\gamma(x) = \frac{s}{\sqrt{d}}[\cos(2\pi x B^T) \| \sin(2\pi x B^T)] \tag{1}$$

where $\|$ is the concatenation of two vectors, and B is a $\frac{d}{2} \times a$ matrix randomly sampled from Gaussian distribution with standard deviation σ. Note that the magnitude (ℓ_2-norm) of $\gamma(x)$ is equal to s. In FFN, the constant $s/\sqrt{d}$ is ignored, hence its magnitude is always $\sqrt{d}$, and σ that controls frequency spectrums is individually searched for different categories. Our network uses a default setting as $s = 80, \sigma = 10$ for all experiments.

Fourier feature embedding facilitates learning complex functions with wider frequency bandwidths, but at the expense of over-fitting issue: introducing unexpected dramatic local variations (e.g., noises, local bumps) to the generated function, especially at unseen spatial positions. As illustrated in Figure 4, at a manifold, the path between two nearby coordinates P_{x_1}, P_{x_2} appears to be overly curvy that would consequently yield unexpected artifacts at spatial positions between them. It happens due to the embedding of the corresponding coordinates x_1 and x_2 with over high frequencies by random Fourier features. The issue motivates us to devise a filter to manipulate the geometry of the functions generated from the network. To this end, our work seeks a new filtering function to mend the reconstruction by enabling better generalization while achieving high fidelity of implicit neural representations.

3.2 Filtering Function

Let $\theta_i, i = 1, 2, ..., k$ denote layers of a $k-$layer MLPs, then the coordinate x passing through the MLPs will produce a sequence of outputs, such as $y_i, i = 1, 2, ..., k$. To reconstruct the target function, we apply supervision on the final output function y_k. And to reduce noisy artifacts and enhance details in y_k, we modulate all the intermediate results of MLPs with a filtering function. It is also possible to filter the final output directly, but more sophisticated designs are needed to trade off the data fidelity and the filtering purpose. Practically, we found that filtering in individual layers of MLPs consequently yields satisfied filtering effects on the final result. The formulation of the new network function can be written in recurrence as follows:

$$\begin{aligned} F_x &= \gamma(x)M^T \\ y_1 &= \theta_1(\gamma(x)) \\ y_i &= o(\theta_i(y_{i-1})) \odot F_x, i = 2, ..., k-1 \\ f_\theta &= \theta_k(y_{k-1}) \end{aligned} \tag{2}$$

where $o(\cdot)$ is the spherical normalization function, such that $o(v) = \frac{v}{\|v\|}$, and $\odot$ is the Hadamard product. Matrix M maps the embedded feature $\gamma(x)$ to the same dimensional of the hidden layers.

The filtering function composed of spherical normalization and Hadamard product derives smoothing and recovering operator respectively that effectively remove the noisy artifacts while maintaining high fidelity of data reconstruction. As illustrated in Figure 4 (left), dramatically local variations occurs between two nearby feature points P_{x_1} and P_{x_2} in an intermediate feature space. After spherical normalization, the path between the normalized features $P^{'}_{x_1}$ and $P^{'}_{x_2}$ becomes flat. This operation yields an over-smoothed reconstruction, as shown in Figure 3(b), that some small structures disappear. Nevertheless, the image is still better than the one without applying a smoothing operation, e.g., Figure 3(a). The smoothing operation indicates that restricting the feature space to hypersphere can still provide both global and local information for data fitting though the results lean toward over-smoothed. To further restore small structures, the recovering operator is applied, and the variations between the new features $P^{''}_{x_1}$ and $P^{''}_{x_2}$ are increased, as shown in Figure 4 (right). The reconstructed image using both operators is shown in Figure 3(c), where fine details are recovered based on Figure 3(b) without introducing more noise.

3.3 Filtering Effects

The filtering effect alters from changing the filtering function, the hyperparameters of the positional encoder, and the number of layers of the MLP network.

Filtering Function Variants. F_x in the filtering function can be devised for different filtering effects. In Figure 5, we compare the visual results of several

Fig. 5. Function F_x is devised to achieve different filtering effects (a-c) by modulating the shape of the feature space. (a) has the smoothest reconstruction without making changes to the feature space. (b), whose feature space is a linear transformed 2D plane multiplied to the hypersphere, is sharper than (a). By increasing more high-frequency variations to the hypersphere brought from $\gamma(x)$, only (c) restores fine-scale details.

variants of F_x, including $F_x = 1, xM^T$ and $\gamma(x)M^T$, that imply a different degree of smoothness. $F_x = 1$ means the recovering operation in FINN is salient, and xM^T is a linear transformation of the original coordinates. The latter slightly increases the geometric variation of the normalized feature space (hypersphere). We use $F_x = \gamma(x)M^T$ in FINN that modulates the shape of the hyperspheres with sinusoidal functions to enable the MLPs to restore high-frequency details.

Why does modulating the geometric variation of feature spaces control the smoothness or sharpness of the result? The intuition is similar to the positional encoding (e.g., sinusoidal [27] or spline [28] based functions) from which coordinates from a smooth space (e.g., 2D plane or 3D cube) are projected to a domain with more complex local variations controlled by some hyperparameters. The main difference is to apply F_x to the intermediate spaces instead of the input domain. Moreover, besides the variants demonstrated, further investigation of F_x for other filtering purposes or more applications would be meaningful.

Hyperparameters. The reconstruction is sensitive to the random Fourier feature embedding, whose parameters are hard to optimize under SGD-based optimization. In Figure 6 (Right), we show the solution space of a testing image by grid-searching the parameters σ (ranging from 1 to 25) and s (ranging from 5 to 160). Specifically, the image is best recovered around $\sigma = 10$ and gets over-smoothed or contaminated by noise when using an overly small or large σ respectively. In contrast to σ, s only slightly change the results when σ is fixed. However, s has more impact on the convergence speed, as shown in Figure 7, where for a fixed σ, faster convergence speed achieves if s grows (from dark to light colors) and becomes stable when $s > 50$. Moreover, higher s also improves network stability, as shown in the highlighted box within the figure where curves are also more linear for $s > 50$. In all our experiments, including image regression and surface reconstruction tasks, we use a constant number of s, i.e., 80. An overly large value of s brings instability to the network, resulting in performance reduction. Note that in FNN, s does not have similar properties as FINN.

Fig. 6. Impact of hyperparameters. (Left) Images are reconstructed using different parameters in positional encoding (σ, s in Equation 1). By increasing σ, image detail becomes clear (a→b), and noise appears when σ is overly high (c). Those images are sampled from the distribution (Right), from which the peak region appears around $\sigma = 10$. As deviated from the peak region, the performance decreases monotonically. The other parameter s has less impact on the image quality if σ is fixed.

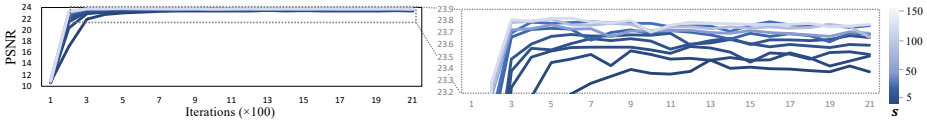


Fig. 7. Convergence speed scales by the magnitude of Fourier feature controlled by parameter s. As s increases the convergence speed grows and reaches stable until $s > 50$. The curves in the highlighted box indicate higher s also improves network stability.

Fig. 8. The influence of the number of hidden layers in FINN. In (1), with only one hidden layer, the reconstructed result still contains notable noisy artifacts. The results are improved when more hidden layers are used (2-5). With more than one hidden layer, our method effectively removes the visual artifacts while keeping small patterns, e.g., in the regions of the sky and ocean. The numerical and visual results are merely changed when the number of the hidden layer exceeds one (2-5).

Number of MLP Layers. Our filtering function performs iteratively in the hidden layers of MLPs. In Figure 8, we showcase the performances of FINN using a different number of hidden layers, ranging from one layer to five layers. With only one hidden layer, the noisy artifacts are still notable in the image, and as the number of hidden layers increases, both visual and numerical results are improved. Besides filtering artifacts, FINN also enhances details in blurry regions. The performance of FINN with only one hidden layer (FINN_{1l}) is already better than state-of-the-art methods, as shown in Table 1. Continuous adding more hidden layers to the network does not produce better results when the number of hidden layers already exceeds two.

4 Experiments

Our network is applicable for the tasks formulated with the implicit function, focusing on representing complex functions that contain a wide range of frequency bandwidths. We first extensively validate the benefits of applying filtering in implicit image functions, then we show the effectiveness of the proposed filtering module in point cloud-based 3D surface reconstruction task.

For image regression, our network contains a random Fourier feature mapping and Multi-Layer Perceptrons (MLPs), as shown in Figure 1. The MLPs have three hidden layers with ReLU activation and a final layer with Sigmoid activation. Each hidden layer is associated with a filtering function, as shown in Figure 1. The dimension is 512 for the Fourier feature and 256 for hidden layers. We use MSE loss for training the images. The network is trained for 2000 epoches with a fixed learning rate of 1e-3. For each image, the training pixels are sampled on a regularly-spaced grid containing 25% of the pixels in the image.

We compare our method to state-of-the-art methods, including FFN [27], SIREN [25], SPE [28] and SAPE [13], on the datasets of natural images and text images from FFN. The error reported are computed on all the pixels in the image, obtained by calculating the similarity between generated images and the corresponding ground truth. We use the standard metrics for similarity measure-

	PSNR↑		ꟻLIP ↓	
	Natural	Text	Natural	Text
FFN	25.57 ± 4.19	30.47 ± 2.11	0.131±0.041	0.096±0.043
SIREN	27.03 ± 4.28	30.81 ± 1.72	0.114±0.040	0.070±0.020
SPE	26.49 ± 3.89	31.12 ± 2.18	0.130±0.038	0.065±0.022
SAPE	28.09 ± 4.04	31.84 ± 2.15	0.118±0.026	0.083±0.041
FINN	**28.51 ± 4.35**	**33.09 ± 1.97**	**0.100±0.037**	**0.042±0.016**
FINN_{1l}	28.08 ± 4.13	32.18 ± 1.81	0.109±0.038	0.053±0.026
FINN_{so}	27.91 ± 4.21	31.19 ± 1.86	0.123±0.036	0.085±0.008

Table 1. Quantitative results on image regression task. FINN_{1l} means FINN with 1 hidden *l*ayer and FINN_{so} means FINN with *s*moothing *o*nly.

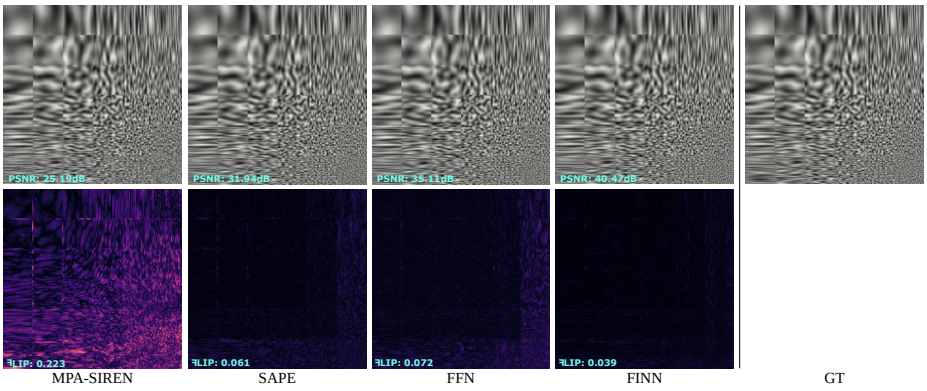


Fig. 9. Reconstruction of Perlin noise functions. A Perlin noise image (right) is synthesized with 6×6 small patches with varying frequencies. FINN reconstructs the image with high quality at all frequencies. MPA-SIREN, SAPE, and FFN are struggling to recover high-frequency components. At low-frequency regions, they also generate notable visual artifacts (better visualized by zooming in).

ment, including PSNR and ꟻLIP [1]. ꟻLIP gives an error map for visualization and a global measurement (i.e., the weighted median of errors on all pixels).

The numerical results are shown in Table 1. Our method outperforms all compared methods with considerable gains. The qualitative results are shown in Figure 12. In general, all the methods can generate realistic images. In particular, FINN can better capture a wider frequency bandwidth than others, doing better in both backgrounds (low-frequency) and small patterns and structures (high-frequency) in the images. In contrast, SAPE generally produces over-smoothed images, while FNN, SIREN, and SPE usually include many noisy artifacts.

Testing with Wide Frequency Bandwidth Image. We demonstrate the capability of our method in reconstructing complex signals containing notably wide frequency bandwidth. Similar to MPA-SIREN [19], a testing image is composed of 6×6 grid of Perlin [23] noise patches with increasing frequencies along with the horizontal and vertical directions. As shown in Figure 9, FINN presents a much better reconstruction compared to MPA-SIREN, SAPE, and FFN. In particular, FINN handles small structures well that other methods usually fail. Even at low-frequency regions, FINN also outperforms compared methods.

Convergence and Stability. Besides achieving better performance, it is surprising that FINN also has good properties in both convergence speed and network stability. Figure 10 shows PSNR curves of all the testing images from the natural and text datasets, where FFN (Left) and FINN (Right) are both trained for 2000 iterations. FINN converges after 300 iterations for most of the images and keeps stable after then. In contrast, FNN is more sensitive to the SGD-based

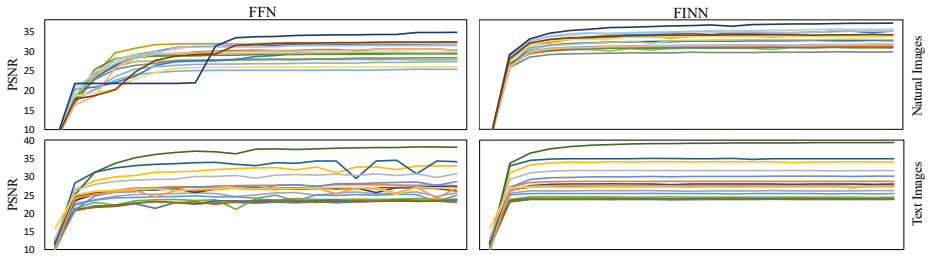


Fig. 10. Compared to FFN (Left), FINN (Right) converges faster and is much more stable. The PSNR curves painted in different colors indicate different testing images from the natural (Top) and text image datasets (Bottom).

optimization, yielding slightly curvy and nonmonotonic testing PSNR curves. The convergence comparison implies that the filtering operation in FINN serves as similar as implicit regularization that resists over-fitting during optimization.

4.1 Applications

Further, we evaluate our method in the 3D surface reconstruction task. We consider the 3D surface as a zero level-set of the signed distance field (SDF) from which every 3D query point has a corresponding value telling its distance and whether inside or outside from the surface. We train the network to fit the SDF on point clouds with additional point normals information. In addition to the SDFs and normals for supervision, we also apply a regularization term to constrain the gradient of the learned function to be a unit vector at any spatial locations, as suggested by [2, 12]. The loss function is defined as

$$\mathcal{L}_{\text{sdf}} = \sum_{x\in\Omega} |\|\nabla f_\theta(x)\|-1| + \sum_{x\in\Omega_0} (|f_\theta(x)| + \|\nabla f_\theta(x) - n(x)\|) + \sum_{x\in\Omega\setminus\Omega_0} \exp(-c|f_\theta(x)|) \tag{3}$$

where Ω is the point set and Ω_0 contains only on-surface points. $f_\theta(x)$ and $\nabla f(x)$ are the fitted SDFs and gradients, and $n(x)$ is the ground truth point normal.

Our network structure and training protocol for surface reconstruction are similar to image regression, except that the final layer becomes a linear transformation and the number of hidden layer increase from 3 to 4. In addition, the dimension of the Fourier feature reduces to 256, and the frequency parameter σ uses one instead of 10 since the manifold of the shape function is commonly much smoother than the image function. Once trained, we uniformly sample a set of regular grid points (i.e., 1600^3) to obtain the corresponding signed distances and use the Marching Cubes algorithm [17] to extract the triangular mesh.

We compare our method with FFN and SIREN on large-scale 3D meshes containing tens of millions of triangles. The testing examples include an object (Left) that has very complex geometric details and a scene (Right) that has both

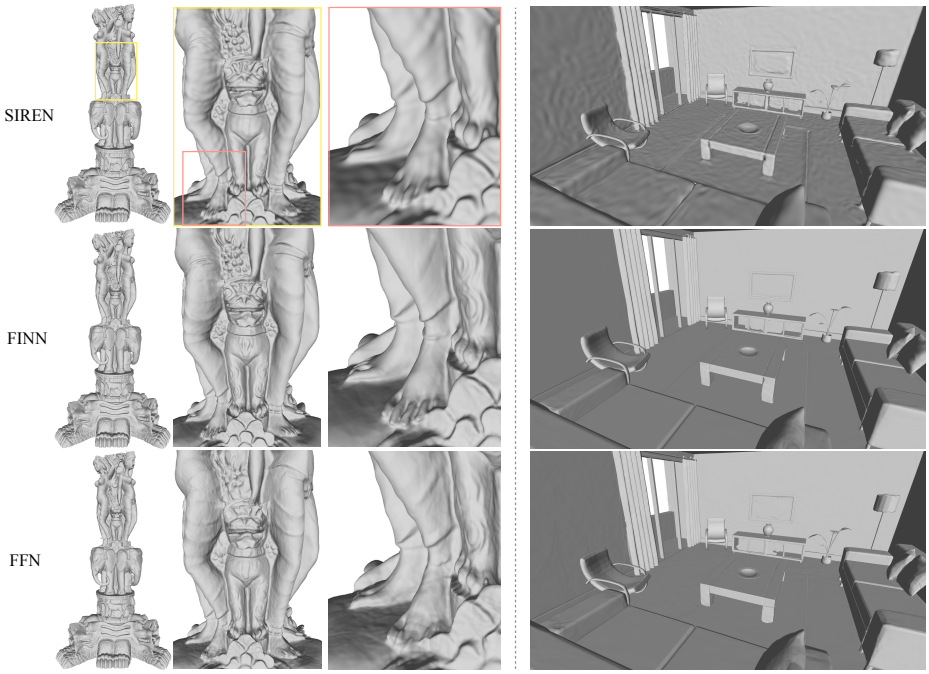
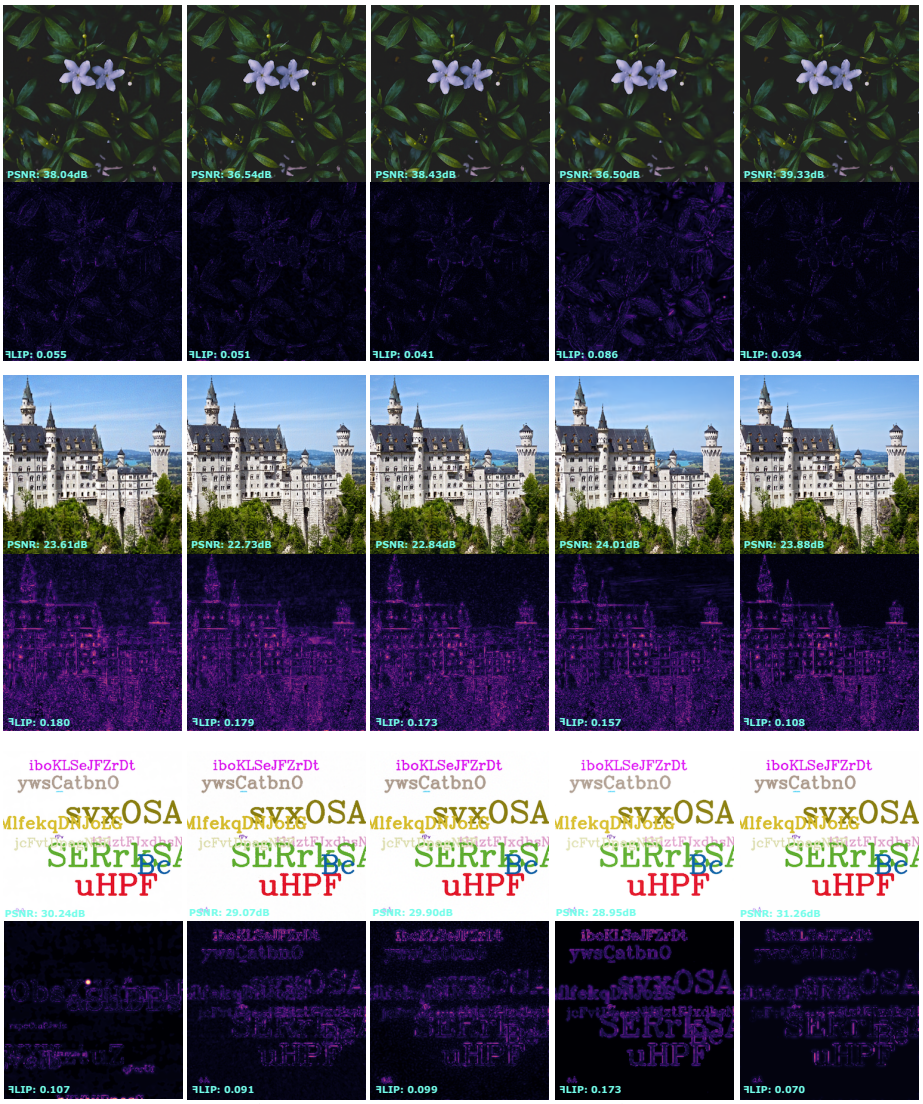


Fig. 11. Qualitative results on surface reconstruction task. Compared to SIREN and FFN, FINN (Middle Row) shows much better reconstruction on both flat regions and sharp surface details.

large flat patches and small objects, as shown in Figure 11. We train FFN and SIREN for 10k epochs and FINN for 5k on the two examples. The reconstruction results show FINN does better in both flat and sharp surface regions. In contrast, FFN and SIREN include more unexpected local bumps and noise and fail to restore many surface details. Similar to image regression, FINN enables the reconstruction of a wide range of frequencies in the target functions, including very flat (low-frequency) regions and sharp (high-frequency) details.

5 Conclusion

We present a novel neural network for implicit functions with explicit consideration of filtering during reconstruction. By integrating a filtering module into the neural network, our method is able to produce complex signals that contain a very wide range of frequency bandwidths. Experimenting on the tasks of 2D image regression and 3D surface reconstruction shows significantly improvement of our method compared to the state-of-the-art methods. Moreover, our filtering technique yields nice properties, such as faster convergence speed and more stable performance under SGD-based optimization.



(a) FFN (b) SPE (c) SIREN (d) SAPE (e) FINN

Fig. 12. Qualitative results on image regression task. Better visualized by zooming in.

Our method shares a similar limitation as FFN [27], i.e., it is sensitive to the global parameters of positional encoding. When using overly high or low frequencies, the generated results become noisy or over-smoothing. Fortunately, parameters by grid-searching are satisfied by most of the examples from the same task without individually tuning. Nevertheless, further investigation into automatic optimization of frequency parameters is needed.

References

1. Andersson, P., Nilsson, J., Akenine-Möller, T., Oskarsson, M., Åström, K., Fairchild, M.D.: ꟻLIP: A Difference Evaluator for Alternating Images. Proceedings of the ACM on Computer Graphics and Interactive Techniques **3**(2), 15:1–15:23 (2020)
2. Atzmon, M., Lipman, Y.: SAL: sign agnostic learning of shapes from raw data. In: CVPR. pp. 2562–2571. Computer Vision Foundation / IEEE (2020)
3. Bemana, M., Myszkowski, K., Seidel, H.P., Ritschel, T.: X-fields: Implicit neural view-, light- and time-image interpolation. ACM Transactions on Graphics (Proc. SIGGRAPH Asia 2020) **39**(6) (2020). https://doi.org/10.1145/3414685.3417827
4. Chabra, R., Lenssen, J.E., Ilg, E., Schmidt, T., Straub, J., Lovegrove, S., Newcombe, R.A.: Deep local shapes: Learning local SDF priors for detailed 3d reconstruction. In: ECCV (29). Lecture Notes in Computer Science, vol. 12374, pp. 608–625. Springer (2020)
5. Chen, Q., Xu, J., Koltun, V.: Fast image processing with fully-convolutional networks. In: ICCV. pp. 2516–2525. IEEE Computer Society (2017)
6. Chen, Y., Liu, S., Wang, X.: 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)
7. Chen, Y., Fernando, B., Bilen, H., Mensink, T., Gavves, E.: Neural feature matching in implicit 3d representations. In: ICML. Proceedings of Machine Learning Research, vol. 139, pp. 1582–1593. PMLR (2021)
8. Chen, Z., Zhang, H.: Learning implicit fields for generative shape modeling. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
9. Fathony, R., Sahu, A.K., Willmott, D., Kolter, J.Z.: Multiplicative filter networks. In: 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net (2021), https://openreview.net/forum?id=OmtmcPkkhT
10. Genova, K., Cole, F., Sud, A., Sarna, A., Funkhouser, T.A.: Local deep implicit functions for 3d shape. In: CVPR. pp. 4856–4865. Computer Vision Foundation / IEEE (2020)
11. Gharbi, M., Chen, J., Barron, J.T., Hasinoff, S.W., Durand, F.: Deep bilateral learning for real-time image enhancement. ACM Trans. Graph. **36**(4), 118:1–118:12 (2017)
12. Gropp, A., Yariv, L., Haim, N., Atzmon, M., Lipman, Y.: Implicit geometric regularization for learning shapes. In: ICML. Proceedings of Machine Learning Research, vol. 119, pp. 3789–3799. PMLR (2020)
13. Hertz, A., Perel, O., Giryes, R., Sorkine-Hornung, O., Cohen-Or, D.: Sape: Spatially-adaptive progressive encoding for neural optimization. In: Thirty-Fifth Conference on Neural Information Processing Systems (2021)
14. Jampani, V., Kiefel, M., Gehler, P.V.: Learning sparse high dimensional filters: Image filtering, dense crfs and bilateral neural networks. In: CVPR. pp. 4452–4461. IEEE Computer Society (2016)
15. Jiang, C.M., Sud, A., Makadia, A., Huang, J., Nießner, M., Funkhouser, T.A.: Local implicit grid representations for 3d scenes. In: CVPR. pp. 6000–6009. Computer Vision Foundation / IEEE (2020)
16. Li, Y., Huang, J., Ahuja, N., Yang, M.: Joint image filtering with deep convolutional networks. IEEE Trans. Pattern Anal. Mach. Intell. **41**(8), 1909–1923 (2019)

17. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3d surface construction algorithm. ACM siggraph computer graphics **21**(4), 163–169 (1987)
18. Martin-Brualla, R., Radwan, N., Sajjadi, M.S.M., Barron, J.T., Dosovitskiy, A., Duckworth, D.: Nerf in the wild: Neural radiance fields for unconstrained photo collections. In: CVPR. pp. 7210–7219. Computer Vision Foundation / IEEE (2021)
19. Mehta, I., Gharbi, M., Barnes, C., Shechtman, E., Ramamoorthi, R., Chandraker, M.: Modulated periodic activations for generalizable local functional representations. In: IEEE International Conference on Computer Vision (ICCV) (2021)
20. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3D reconstruction in function space. In: Proc. CVPR (2019)
21. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: ECCV (1). Lecture Notes in Computer Science, vol. 12346, pp. 405–421. Springer (2020)
22. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: DeepSDF: Learning continuous signed distance functions for shape representation. In: CVPR (2019)
23. Perlin, K.: Improving noise. ACM Trans. Graph. **21**(3), 681–682 (2002)
24. Saito, S., Huang, Z., Natsume, R., Morishima, S., Kanazawa, A., Li, H.: Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In: International Conference on Computer Vision. pp. 2304–2314 (2019)
25. Sitzmann, V., Martel, J.N., Bergman, A.W., Lindell, D.B., Wetzstein, G.: Implicit neural representations with periodic activation functions. In: Proc. NeurIPS (2020)
26. Sitzmann, V., Zollhöfer, M., Wetzstein, G.: Scene representation networks: Continuous 3d-structure-aware neural scene representations. In: NeurIPS
27. Tancik, M., Srinivasan, P.P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J.T., Ng, R.: Fourier features let networks learn high frequency functions in low dimensional domains. NeurIPS (2020)
28. Wang, P., Liu, Y., Yang, Y., Tong, X.: Spline positional encoding for learning 3d implicit signed distance fields. In: IJCAI. pp. 1091–1097. ijcai.org (2021)
29. Williams, F., Schneider, T., Silva, C.T., Zorin, D., Bruna, J., Panozzo, D.: Deep geometric prior for surface reconstruction. In: CVPR. pp. 10130–10139. Computer Vision Foundation / IEEE (2019)
30. Williams, F., Trager, M., Bruna, J., Zorin, D.: Neural splines: Fitting 3d surfaces with infinitely-wide neural networks. In: CVPR. pp. 9949–9958. Computer Vision Foundation / IEEE (2021)
31. Xu, Q., Wang, W., Ceylan, D., Mech, R., Neumann, U.: DISN: deep implicit surface network for high-quality single-view 3d reconstruction. In: NeurIPS. pp. 490–500 (2019)
32. Xu, Y., Fan, T., Yuan, Y., Singh, G.: Ladybird: Quasi-monte carlo sampling for deep implicit field based 3d reconstruction with symmetry. In: ECCV (1). Lecture Notes in Computer Science, vol. 12346, pp. 248–263. Springer (2020)
33. Yoon, Y., Jeon, H., Yoo, D., Lee, J., Kweon, I.S.: Learning a deep convolutional network for light-field image super-resolution. In: ICCV Workshops. pp. 57–65. IEEE Computer Society (2015)
34. Yu, A., Ye, V., Tancik, M., Kanazawa, A.: pixelNeRF: Neural radiance fields from one or few images. In: CVPR (2021)
35. Zhang, J., Pan, J., Lai, W., Lau, R.W.H., Yang, M.: Learning fully convolutional networks for iterative non-blind deconvolution. In: CVPR. pp. 6969–6977. IEEE Computer Society (2017)
36. Zhi, S., Laidlow, T., Leutenegger, S., Davison, A.: In-place scene labelling and understanding with implicit scene representation. In: Proceedings of the International Conference on Computer Vision (ICCV) (2021)