Volume Upscaling with Convolutional Neural Networks

Zhenglei Zhou, Yule Hou, Qirui Wang, Guangxiang Chen, Jiawei Lu, Yubo Tao *
and Hai Lin
State Key Lab of CAD&CG, Zhejiang University

ABSTRACT

Volume upscaling generates high-resolution volumes from low-resolution volumes to make data exploration more effective. Traditional methods, such as the simple trilinear or cubic-spline interpolation, may blur boundaries of features and lead to jagged artifacts. Inspired by recent progress in image super-resolution with Convolutional Neural Networks (CNN), we propose a CNN-based volume upscaling method. Our CNN contains three hidden layers: block extraction and representation, non-linear mapping, and reconstruction. It directly learns an end-to-end mapping from low-resolution blocks to high-resolution volume. Compared to previous methods, our CNN can preserve better structures and details of features, and provide a better volume quality in both the visualization and evaluation metrics.

CCS CONCEPTS

• **Human-centered computing** → **Scientific visualization**; • **Computing methodologies** → *Neural networks*;

KEYWORDS

Volume, Upscaling, Convolutional neural network

1 INTRODUCTION

Volume visualization has been widely used to explore volume datasets in various fields, such as biology, medicine, geology, and climate. With the development of digital imaging techniques, high-resolution volume datasets can be acquired more easily in several modalities, such as Computed Tomography (CT). However, there are still many low-resolution digital imaging devices, which generates low-resolution volume datasets continuously. In the exploration process, these volumes are often upscaled implicitly in the resampling and reconstruction stage during direct volume rendering, especially when the user zooms in on features of interest in the volume. The high-resolution reconstruction usually employs the simple trilinear or cubic-spline interpolation in order to provide a high rendering efficiency, and the scaling factor depends on the sampling step and zooming ratio. These interpolations are only based on local information around the interpolated position, and therefore may blur boundaries of features and result in jagged artifacts in the rendered image. In fact, besides implicit volume upscaling on demand during the rendering, volume datasets can be explicitly upscaled offline to construct high-resolution volume datasets for analysis and visualization. In this way, we can employ more relevant information and complex construction methods to improve the quality of the enlarged volume.

Explicit volume upscaling methods have been studied recently in volume visualization. Grevera and Udupa [9] compared 3D image interpolation methods, especially in the across-slice direction of medical datasets due to their non-homegeneous resolutions. Giachetti et al. [8] proposed an edge adaptive and energy preserving volume upscaling method, and improved the volume upscaling quality of medical datasets. In image upscaling, state-of-the-art methods are largely based on the example-based strategy [17]. The internal example-based methods [5, 6] exploit internal similarities of patches in the same image, and recover the missing high-frequency components in the upscaled image. Wang et al. [16] further validated the local self-similarity assumption for 3D volumes on small scaling factors, and extended this technique to slice upscaling and 3D volume upscaling. The external example-based methods [11, 18] mostly learn dictionaries from external image datasets to model the patch space. Recently, Dong et al. [4] demonstrated that the sparse-coding-based method, one representative external example-based method, can be viewed as one deep convolutional neural network, and proposed a deep learning method to directly learn an end-to-end mapping from the low-resolution image to the high-resolution image. Compared to state-of-the-art example-based methods, this method produces a better image quality. This external example-based method has not been exploited in explicit volume upscaling.

In this paper, we propose a convolutional neural network (CNN) based method for explicit volume upscaling, which directly learns an end-to-end mapping from the low-resolution volume dataset to the high-resolution volume dataset. The structure of our CNN is inspired from the sparse coding framework and the image super-resolution convolutional neural network [4]. It contains three hidden layers: block extraction and representation, non-linear mapping, and reconstruction. The volumes in the train set are first downsampled as the low-resolution volumes. Then, we train our CNN by minimizing the mean squared error between the enlarged volume from the downsampled volume and the original volume.

*Corresponding author: Yubo Tao (taoyubo@cad.zju.edu.cn)

CGI '17, June 27-30, 2017, Yokohama, Japan
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-5228-4/17/06...$15.00
https://doi.org/10.1145/3095140.3095178

We have experimented different configurations of hyper-parameters in our CNN to search for the optimal hyper-parameters for volume upscaling. Compared to previous volume upscaling methods, our method can provide a better volume quality in both the visualization and evaluation metrics, such as PSNR (Peak Signal-to-Noise Ratio) and SSIM (Structural Similarity Index), which is proved to be consistent with visual human perception and better than PSNR.

2 RELATED WORK

There are many researches on volume upscaling. Medical volume datasets that are largely generated by digital imaging devices are usually not in a homogeneous resolution, and have a coarser resolution between slices than within slices. Z-upscaling [9] has been used to increase the across-slice resolution of medical datasets. Grevera and Udupa [9] summarized these interpolation methods that extract information from neighbor images to guide the interpolation. Besides slice upscaling, volume datasets can be enlarged during the volume construction stage directly from the raw data produced by digital imaging devices. A frequency-domain upsampling method based on a Body-Centered Cubic lattice has been proposed by Csébfalvi and Domonkos [2] to improve the rendering quality.

Volume upscaling can be performed after the construction but before volume rendering. Giachetti et al. [8] extended the irradicance preserving image interpolation [7] to upscale volumes using voxel subdivision and refinement. Wang et al. [16] applied the idea of the internal example-based methods to isotropic volume upscaling. This method predicts the missing high-resolution detail from other example blocks of the original volume. In this paper, our volume upscaling method is based on the external example-based methods, which have the state-of-the-art performance. In addition, our method is not limited to the integer or $N + 1 : N$ scaling factor, and we can upscale volumes by any arbitrary factor.

Our method is based on convolutional neural networks (CNN), which was first proposed by LeCun et al. [14] in 1989. CNN, a type of neural networks, consist of a number of layers, especially convolutional layers for local connections, which reduce the number of parameters. The parameters of each layer are learned to optimize performance on the training set. Since AlexNet [13] demonstrated impressive performance on the ImageNet 2012 classification benchmark, the CNNs have achieved great success in computer vision, speech recognition, and natural language processing, and shown state-of-the-art performance on various tasks [1]. The success of CNN can be explained by powerful GPU implementations, larger training sets, and better models with powerful representations. Recently, Dong et al. [3, 4] discussed the relation between the sparse-coding-based method and convolutional neural networks, and proposed a deep learning method to optimize the whole pipeline in sparse-coding-based methods and revealed its feasibility in image upscaling. Based on CNN, our volume upscaling method also optimizes all steps in the mapping between low/high-resolution volume datasets. To the best of our knowledge, this is the first paper to apply CNN to volume upscaling in visualization. However, it is challenging to directly apply CNN to 3D volumes, since the number of parameters of CNN increases dramatically from 2D to 3D.

3 CNN BASED VOLUME UPSCALING

The sparse-coding-based method has been applied to solve various reconstruction tasks, e.g., denoising, impainting, and image super-resolution. Generally, it consists of three steps for image super-resolution. Firstly, it projects each low-resolution patch extracted from the input image onto the low-resolution dictionary. Secondly, it iteratively performs non-linear mapping to project the above sparse coefficients to the high-resolution dictionary, and the output are the sparse coefficients of high-resolution patches. Lastly, it reconstructs the high-resolution patch through linear combination and averaging the overlapping high-resolution patches.

As demonstrated in [4], the sparse-coding-based method can be viewed as a type of convolutional neural network. The convolutional neural network optimizes an end-to-end mapping that consists of three steps above. We apply the convolutional neural network to implement explicit volume upscaling. As shown in Figure 1, our CNN similarly contains three hidden layers. We first crop low-resolution blocks from the input volume data, and convolve them with the first layer's neurons to extract feature maps from blocks. The second layer leverages generated feature maps to perform non-linear mapping, and the output is the representation of high-resolution blocks. The third layer reconstructs the high-resolution volume. The loss between the output and ground truth volume is then computed to tune the parameters of the network, and the optimization goal is to make the loss in the training set as lower as possible and the network generalization ability as stronger as possible. We describe our convolutional neural network for volume upscaling and how to train the network using 3D blocks in the following subsection.

3.1 Network Architecture

Convolution Neural Networks (CNN) nowadays plays a significant role in many research fields, such as classification, recognization, and even 3D tasks. In fact, CNN is very similar to ordinary Neural Network. Generally, CNN takes multiple Convolutional Layers to extract features, and then these features feed the next fully-connected layer to solve domain-specific problems. Its superior performance and generalization ability mainly takes advantage of the properties of the convolutional Layer:

Local Connectivity: Instead of connecting neurons to all neurons in the previous layer, the convolutional layer connects each neuron to only a local region of the previous layer. We call the local region as the local receptive field, and the idea dates back to the 1960s with the perceptron and Hubel and Wiesel's [10] discovery of locally sensitive. In this way, the number of parameters is decreased and this architecture is more practical.

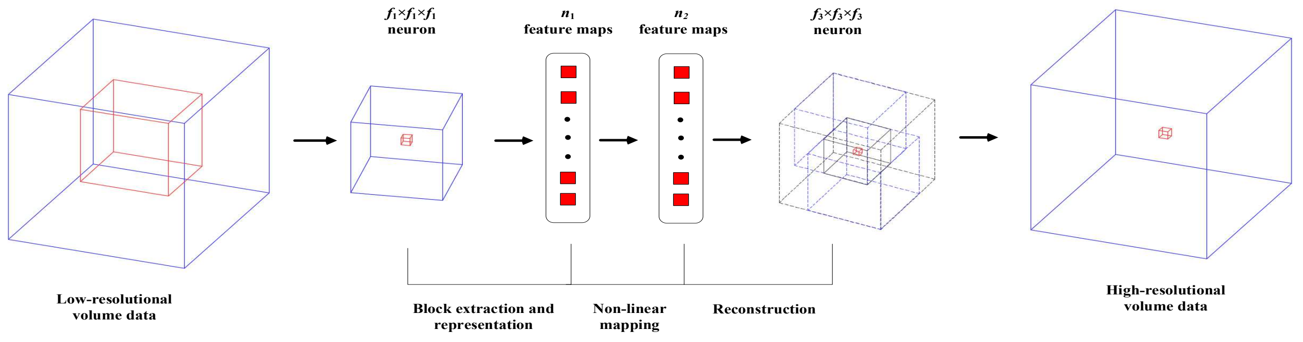
$f_1 \times f_1 \times f_1$ neuron
n_1 feature maps
n_2 feature maps
$f_3 \times f_3 \times f_3$ neuron
Low-resolutional volume data
Block extraction and representation
Non-linear mapping
Reconstruction
High-resolutional volume data

Figure 1: The pipeline of our CNN. We first convolve the cropped $f_1{\times}f_1{\times}f_1$ block with n_1 neurons of support $f_1{\times}f_1{\times}f_1$ to extract n_1 activations. n_2 activations are then generated after non-linear mapping, and we crop $n_2{\times}f_3{\times}f_3{\times}f_3$ blocks to reconstruct the high-resolution volume.

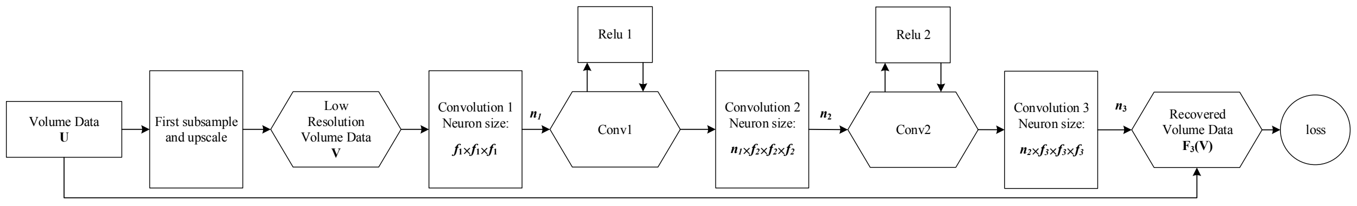
Volume Data U
First subsample and upscale
Low Resolution Volume Data V
Convolution 1 Neuron size: $f_1 \times f_1 \times f_1$
n_1
Relu 1
Conv1
Convolution 2 Neuron size: $n_1 \times f_2 \times f_2 \times f_2$
n_2
Relu 2
Conv2
Convolution 3 Neuron size: $n_2 \times f_3 \times f_3 \times f_3$
n_3
Recovered Volume Data F3(V)
loss

Figure 2: The network architecture of our CNN. The CNN consists of three convolutional layers, and the loss measurement is Euclidean Loss.

Parameter Sharing: If one patch feature is useful to compute at one spatial position, it should also be useful to compute at another position. The parameter sharing scheme is used in the convolutional layer to control the number of parameters and also avoids overfitting.

The proposed CNN contains three convolutional layers as shown in Figure 2, and there are three hyper-parameters, including depth, stride and padding, to control the size of the output volume. The depth hyper-parameter specifies how many neurons in the convolutional layer connect to the same region of the input volume, and the stride hyper-parameter spatially controls the size of output volumes. By taking a small stride, we crop blocks densely from the input volume, and thus obtain large output volumes. The padding hyper-parameter, especially zero-padding, is used to preserve the spatial size of the input volume. In our CNN, we refer to five parameters, f_1, f_2, f_3, n_1 and n_2, as the neuron size of three layers and the neuron number of the first two layers.

In the training process, the initial volume is denoted as $\mathbf{U}$, and it is sub-sampled and upscaled to the same size via cubic interpolation to generate the input volume $\mathbf{V}$. The proposed CNN is supposed to recover the output volume $F(\mathbf{V})$ from $\mathbf{V}$, and the difference between the initial volume $\mathbf{U}$ and the output volume $F(\mathbf{V})$ should be as small as possible. The three convolutional layers in Figure 2 are described as follows:

Block extraction and feature representation: The first layer extracts blocks from the input volume $\mathbf{V}$ and represents each block as a high-dimensional vector. Formally, this layer is expressed as an operation F_1:

$$F_1(\mathbf{V}) = max(0, W_1 * \mathbf{V} + B_1). \tag{1}$$

W_1 corresponds to n_1 neurons with the size $f_1{\times}f_1{\times}f_1$, where f_1 is the spatial size of a neuron, and n_1 is the number of neurons. B_1 is a n_1-dimensional biases vector, and '$*$' denotes a convolution operation. The output are n_1 activations. We apply the Rectified Linear Unit (ReLU, max(0; x)) [15] for faster convergence while preserving good performance.

Non-linear mapping: The second layer nonlinearly maps each high-dimensional vector onto another high-dimensional vector. For instance, our first layer generates n_1-dimensional vectors, and they are fed to the second layer. The operation is described as follows:

$$F_2(\mathbf{V}) = max(0, W_2 * F_1(\mathbf{V}) + B_2). \tag{2}$$

W_2 consists of n_2 neurons with the size $n_1{\times}f_2{\times}f_2{\times}f_2$, and B_2 is a n_2-dimensional biases vector. A n_2-dimensional vector represents a high-resolution block, and they are used to in the third layer to reconstruct the output volume.

Reconstruction: The third layer aggregates above high-resolution blocks to generate the final volume. We define this procedure as follows:

$$F_3(\mathbf{V}) = W_3 * F_2(\mathbf{V}) + b. \tag{3}$$

W_3 is a neuron with the size $n_2{\times}f_3{\times}f_3{\times}f_3$, and b is a bias. The output volume $F(\mathbf{V})$ is supposed to be similar to the initial volume $\mathbf{U}$.

Based on these three convolutional layers, our CNN learns an end-to-end mapping and both weights and biases are optimized. Furthermore, the proposed CNN leverages more neighboring voxels, and consequently our CNN achieves better performance. For example, we assume that the neuron size $f_1 = 5$, $f_2 = 1$ and $f_3 = 3$. On the whole, the estimation of a high-resolution voxel uses the information of $(5 + 3 - 1)^3 = 343$ voxels.

3.2 Network Training

Since the sizes of available volumes are different, we first crop these volumes into blocks with the same size for computation simplicity. Specifically, each volume is cropped into multiple $f_{sub} \times f_{sub} \times f_{sub}$ blocks based on the stride, and each block is the initial volume in the training process.

CNN can obtain better performance from big training data, as more different data can avoid overfitting in the CNN training. In our case, many blocks are similar after cropping from different volumes. For example, many blocks are empty with the zero scalar value, and we can only preserve part of them to train our CNN with the similar performance. In addition, the GPU memory is limited and we could not use all blocks for training. Thus, it is helpful to select different blocks under the memory limit to train our CNN for volume upscaling. Blocks are first grouped based on their scalar values, and we remove duplicated blocks in each group. The number of preserved blocks in each group is proportional to the number of blocks in each group. As a result, more popular blocks will be trained more times. After removing similar blocks, we can obtain a training set with more different blocks under the same size of GPU memory.

Since the ground truth volume is generally difficult to obtain for available volumes, each block is sub-sampled and upscaled to the same size of the block as the input volume $\mathbf{V}$ and the block is considered as the ground truth volume $\mathbf{U}$. Each input volume $\mathbf{V}_i$ is fed to our CNN with three convolutional layers to compute the loss with the corresponding ground truth volume $\mathbf{U}_i$. The loss is then propagated back through the layers to adjust the parameters to achieve volume upscaling effectively. More concretely, our CNN learns an end-to-end mapping to make the difference between the reconstructed volume $F(\mathbf{V}_i)$ and the ground truth volume $\mathbf{U}_i$ as small as possible. The parameter set of our CNN is $\Theta = \{W_1, W_2, W_3, B_1, B_2, B_3\}$, and the loss function with regard to Θ is

$$L(\Theta) = \frac{1}{N}\sum_{i=1}^{N} \|F(\mathbf{V}_i;\Theta) - \mathbf{U}_i\|^2, \tag{4}$$

where N is the number of blocks in the training set. The MSE (Mean Squared Error) loss function is used in our training, since it is a widely-used metric for quantitatively evaluating the volume reconstruction quality, and the low loss favors a high PSNR, which is most commonly used to measure the quality of reconstruction of lossy compression codecs.

In order to avoid border effects, our convolutional layers have no padding. Therefore, the output of our CNN produces a smaller $(f_{sub} - f_1 - f_2 - f_3 + 3)^3$ volume, and the loss function only compares it with the central part of the ground truth volume. The loss is minimized using stochastic gradient descent with the standard back-propagation. The mechanism of updating network parameter is

$$\Delta_{i+1} = 0.9 \cdot \Delta_i + \eta \cdot \frac{\partial L}{\partial W_i^l}, \quad W_{i+1}^l = W_i^l + \Delta_{i+1}, \tag{5}$$

where $l \in \{1, 2, 3\}$ indicates the l-th layer, i is the iteration number, and $\frac{\partial L}{\partial W_i^l}$ is derivative. The neuron weights (parameters) of each layer are initialized by drawing randomly from a Gaussian distribution with zero mean and standard deviation 10^{-3}. Besides, the biases are from scratch (0). η is the learning rate. We set η of the first two layer to be 10^{-3}, and the third layer to be 10^{-4}. The momentum is 0.9 to avoid turbulence. If the loss grows turbulent, we decrease η by a fact of 10.

We implement our CNN with the *Caffe* package [12] and experiment several volume datasets to demonstrate its efficiency.

4 RESULTS

Our dataset are mainly from online accessible volume data library, such as volvis[1] and vollib[2]. We experiment on noisy training data, and discover that the CNN learn the feature of noises and reconstruct with noises when testing since our ground truth volumes $\mathbf{U}$ contain noises. Thus, we avoid selecting volumes with a high noise level as the training data.

Table 1: Training set and testing set. train* is used in the network architecture comparison.

Volume Name	Spatial Size	Train/Test
Visible Male	256*256*128	train*
Tooth	256*256*161	train*
Colon Supine	512*512*426	train*
Cadaver Head	512*512*106	train*
Registered Monkey Head	256*256*62	train
Foot	256*256*256	train
Female Chest	384*384*240	train
Colon Prone	512*512*463	test
Baby Head	256*256* 98	test
Chapel Hill CT Head	256*256*113	test

We compare several network architectures with an upsacling factor of 2 from different aspects: neuron size, neuron number, network depth, and block size. We first exploit the best architecture for each block size (f_{sub}), and then compare these best architectures to choose the best block size for our network architecture. During network architecture comparison, the training set and testing set consist of four and three CT (Computed Tomography) volume datasets, respectively, as shown in the first four rows and last three rows of Table 1. In order to learn more features and make full use of the training set, we adopt a smaller stride in the training set than the one in the testing set. Taking a stride of 5, 7 and 10

[1] http://www.volvis.org/
[2] http://www9.informatik.uni−erlangen.de/External/vollib/

Table 2: The Average PSNR and SSIM by using different methods on all CT volume datas. Our CNN has the highest Average PSNR or SSIM.

Methods	Trilinear	Cubic-spline	our CNN
PSNR	37.25	38.78	**41.16**
SSIM	0.990	0.992	**0.995**

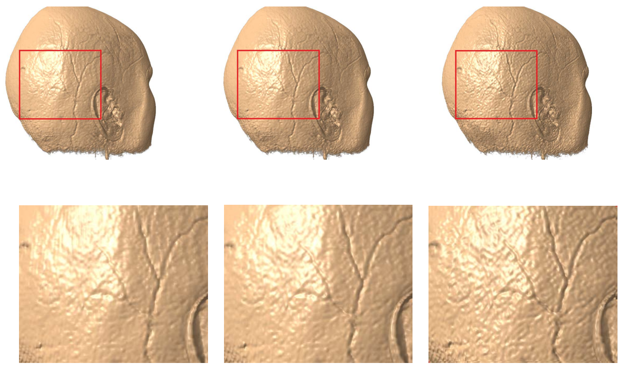
(a) trilinear (b) cubic-spline (c) our CNN

Figure 4: The head dataset is directly upscaled by trilinear interpolation (a), cubic-spline interpolation (b), and our CNN (c). Our result shows much more details on the surface of the head.

for each cropped block size of 12, 16 and 20 in the training set, we get 1 170 000, 409 472 and 139 904 corresponding training blocks. For the testing set, we take a stride of 8, 10 and 12 for each cropped block size of 12, 16 and 20, and then our testing blocks comprise 124 648, 62 187 and 35 634 blocks, respectively. Considering the trade-off between speed and performance, we finally adopt the network architecture (f_1=7, f_2=1, f_3=3, n_1=64, n_2=32 and f_{sub}=16) as the basic setting for our proposed CNN.

With the basic setting, we further train our ultimate model with seven volume datasets. Every 100 generations of the training process takes around 14 seconds on NVIDIA GeForce GTX 980Ti GPU, and the loss quickly get stable after around 80 000 generations. Although our training set only contains a small number of volumes, we believe that the number of cropped blocks is large enough to learn an end-to-end mapping for volume upscaling. Moreover, we test the basic network architecture on 34 CT volume datasets. As shown in Table 2, our CNN shows a higher average PSNR or SSIM and achieves a better performance, compared with trilinear interpolation and cubic-spline interpolation.

We visually compare the upscaled results generated by our CNN with the ones by previous methods in Figure 3-5. Since it is nearly impossible to obtain the ground truth upscaled volume, we first downsample the cadaver head dataset to simulate the low resolution acquisition. The downsampled volume is upscaled with the scaling factor of 2 by trilinear interpolation, cubic-spline interpolation, and our CNN in

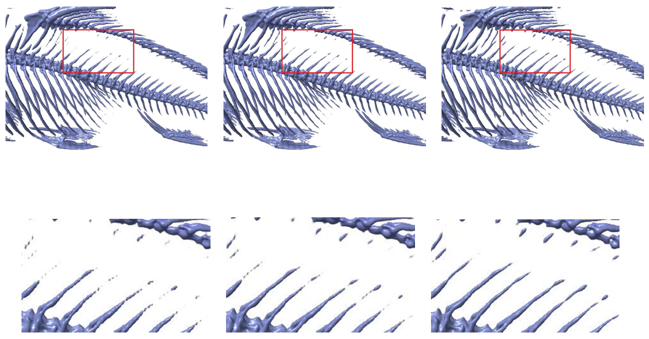
(a) trilinear (b) cubic-spline (c) our CNN

Figure 5: The carp dataset is directly upscaled by trilinear interpolation (a), cubic-spline interpolation (b), and our CNN (c). Our result enhances the continuity of the fish bone more than other methods.

Figure 3. It is clear that our result reduces the jagged artifacts on the face and enhances features near the eye than the other two methods. Thus, our result is most similar to the original volume. We also directly upscale three different volumes by 2, and compare our result with trilinear interpolation and cubic-spline interpolation. Our result of the head volume in Figure 4 shows much more details on the surface of the head. Compared with the results of trilinear interpolation and cubic-spline interpolation, the continuity of the fish bone in Figure 5 are enhanced more clearly.

We further compare our method with self-example based method [16] using the kidney volume in Figure 6. As self-example based method [16] can only upscale volumes by $N+1:N$ scaling factor, the scaling factor in Figure 6 is 1.5 ($N=2$). Although self-example based method [16] enhances the structures of strong vessels, some week vessels as indicated in the box are only enhanced by our method. From these visual comparisons, our result shows much more details and clearer structures in the visualization. In addition, the PSNR of the kidney volume by self-example based method is 39.71 dB, while the PSNR of our result is 41.71 dB.

CT volume datasets differ significantly from other types of data, such as MRI and PET, and it is better to train different CNN for different types. To verify this assumption, we train a new model with four MRI volume datasets under the same configuration of the best network architecture. This new model obtains 0.39 dB higher average PSNR than the model trained by CT volume datasets when testing on ten MRI volume datasets. Thus, different models should be trained to achieve a better performance on different types of volumes.

5 CONCLUSION

We have proposed a CNN-based volume upscaling method to learn an end-to-end mapping from low-resolution volumes to high-resolution volumes. Compared to other methods, experiments demonstrated that our method can generate a better volume quality in both the visualization and evaluation

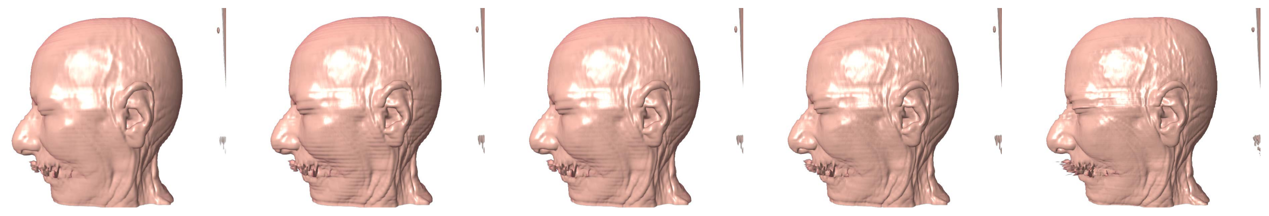
(a) downsampled (b) trilinear (c) cubic-spline (d) our CNN (e) original

Figure 3: The cadaver head dataset is first downsampled in (a), and then upscaled by trilinear interpolation (b), cubic-spline interpolation (c) and our CNN (d). Compared with the original data (e), our result reduces the jagged artifacts on the face and enhances features near the eye.

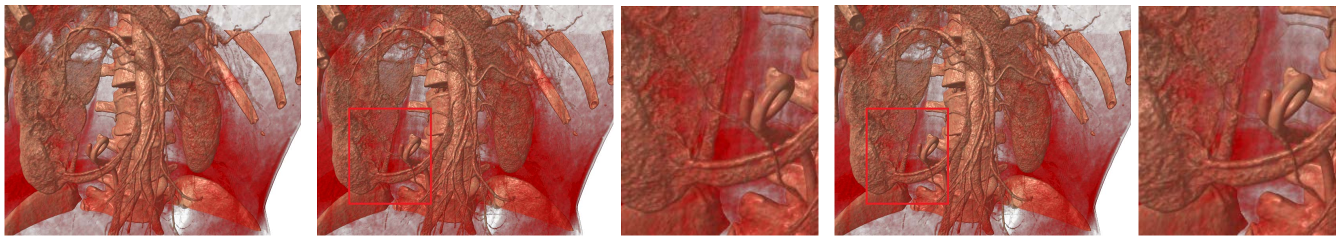
(a) original (b) our CNN (c) self-examples [16]

Figure 6: The kidney dataset (a) is directly upscaled by our CNN (b) and self-example based method [16] (c). Compared with the result in (c), our result enhances the vessels more clearly.

metrics, such as PSNR and SSIM. In the future, we plan to visually display the neurons and their activations on volumes of our learned CNN model and exploit deeper neural networks to enhance the performance of volume upscaling.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments. This work was partially supported by National Natural Science Foundation of China No. 61472354, 61672452 and U1611263.

REFERENCES

[1] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. 2014. Return of the Devil in the Details: Delving Deep into Convolutional Nets. In *BMVC 2014*.

[2] B. Csébfalvi and B. Domonkos. 2009. Frequency-Domain Upsampling on a Body-Centered Cubic Lattice for Efficient and High-Quality Volume Rendering. In *VMV*. 225–232.

[3] C. Dong, C. Loy, K. He, and X. Tang. 2014. Learning a Deep Convolutional Network for Image Super-Resolution. In *ECCV 2014*. Lecture Notes in Computer Science, Vol. 8692. 184–199.

[4] C. Dong, C. Loy, K. He, and X. Tang. 2016. Image Super-Resolution Using Deep Convolutional Networks. *TPAMI,* 38, 2 (2016), 295–307.

[5] M. Ebrahimi and E. Vrscay. 2007. Solving the inverse problem of image zooming using "self-examples". In *ICIARąŕ07*. 117–130.

[6] G. Freedman and R. Fattal. 2011. Image and Video Upscaling from Local Self-Examples. *TOG,* 30, 2 (2011), 12.

[7] A. Giachetti. 2010. Irradiance Preserving Image Interpolation. In *ICPR 2010*. 2218–2221.

[8] A. Giachetti, J. A. I. Guitián, and E. Gobbetti. 2010. Edge Adaptive and Energy Preserving Volume Upscaling for High Quality Volume Rendering. In *EG-IT 2010*, Vol. 10. The Eurographics Association, 17–23.

[9] G.J. Grevera and J.K. Udupa. 1998. An objective comparison of 3-D image interpolation methods. *T-MI,* 17, 4 (1998), 642–652.

[10] D. H. Hubel and T. N. Wiesel. 1962. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology* 160, 1 (1962), 106.

[11] K. Jia, X. Wang, and Tang X. 2013. Image Transformation Based on Learning Dictionaries across Image Spaces. *TPAMI,* 35, 2 (2013), 367–380.

[12] Y. Jia, J. Shelhamer, E., S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. 2014. Caffe: Convolutional architecture for fast feature embedding. In *ACM MM*. 675–678.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS 2012*. 1097–1105.

[14] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. 1989. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Comput.* 1, 4 (Dec. 1989), 541–551.

[15] V. Nair and G. E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML-10*. 807–814.

[16] Q. Wang, Y. Tao, C. Wang, F. Dong, H. Lin, and G. Clapworthy. 2013. Volume Upscaling Using Local Self-Examples for High Quality Volume Visualization. In *CAD/Graphics 2013*. 298–305.

[17] C. Y. Yang, C. Ma, and M. H. Yang. 2014. Single-Image Super-Resolution: A Benchmark. In *ECCV 2014*. 372–386.

[18] J. Yang, J. Wright, T. S. Huang, and Y. Ma. 2010. Image Super-resolution via Sparse Representation. *IEEE Transactions on Image Processing,* 19, 11 (Nov. 2010), 2861–2873.