Voxel Deconvolutional Networks for 3D Brain Image Labeling

Yongjun Chen
Washington State University
Pullman, WA, USA
yongjun.chen@wsu.edu

Hongyang Gao
Washington State University
Pullman, WA, USA
hongyang.gao@wsu.edu

Lei Cai
Washington State University
Pullman, WA, USA
lei.cai@wsu.edu

Min Shi
Washington State University
Pullman, WA, USA
min.shi@wsu.edu

Dinggang Shen
University of North Carolina
Chapel Hill, NC, USA
dgshen@med.unc.edu

Shuiwang Ji
Washington State University
Pullman, WA, USA
sji@eecs.wsu.edu

ABSTRACT

Deep learning methods have shown great success in pixel-wise prediction tasks. One of the most popular methods employs an encoder-decoder network in which deconvolutional layers are used for up-sampling feature maps. However, a key limitation of the deconvolutional layer is that it suffers from the checkerboard artifact problem, which harms the prediction accuracy. This is caused by the independency among adjacent pixels on the output feature maps. Previous work only solved the checkerboard artifact issue of deconvolutional layers in the 2D space. Since the number of intermediate feature maps needed to generate a deconvolutional layer grows exponentially with dimensionality, it is more challenging to solve this issue in higher dimensions. In this work, we propose the voxel deconvolutional layer (VoxelDCL) to solve the checkerboard artifact problem of deconvolutional layers in 3D space. We also provide an efficient approach to implement VoxelDCL. To demonstrate the effectiveness of VoxelDCL, we build four variations of voxel deconvolutional networks (VoxelDCN) based on the U-Net architecture with VoxelDCL. We apply our networks to address volumetric brain images labeling tasks using the ADNI and LONI LPBA40 datasets. The experimental results show that the proposed iVoxelDCNa achieves improved performance in all experiments. It reaches 83.34% in terms of dice ratio on the ADNI dataset and 79.12% on the LONI LPBA40 dataset, which increases 1.39% and 2.21% respectively compared with the baseline. In addition, all the variations of VoxelDCN we proposed outperform the baseline methods on the above datasets, which demonstrates the effectiveness of our methods.

CCS CONCEPTS

• **Theory of computation** → **Structured prediction**; • **Computer systems organization** → **Neural networks**; • **Computing methodologies** → Artificial intelligence;

KEYWORDS

Deep learning, voxel deconvolutional layer, voxel deconvolutional networks, volumetric brain image labeling

ACM Reference Format:
Yongjun Chen, Hongyang Gao, Lei Cai, Min Shi, Dinggang Shen, and Shuiwang Ji. 2018. Voxel Deconvolutional Networks for 3D Brain Image Labeling. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3219819.3219974

1 INTRODUCTION

In recent years, deep learning methods play an important role in various computer vision tasks such as image classification [14], data completion [17], action recognition [12] and pixel-wise prediction [4, 11, 26]. Some key network layers like convolutional layers [15], pooling/unpooling layers [21, 34], deconvolutional layers [28] and some well-known models like generative models [9] and encoder-decoder architectures [21, 23] are frequently used for these tasks. In pixel-wise prediction tasks, U-Net [23] with an encoder-decoder architecture is commonly used. The encoder path contains convolutional and down-sampling operations to extract high-level feature maps from raw images, while the decoder path recovers feature maps to the original spatial size using up-sampling operations. The architecture of an encoder is very similar to a classic convolutional neural networks which have been studied extensively [16, 23]. On the other hand, less attention is paid to the decoder path. There are three primary ways to up-sample feature maps; namely deconvolution, unpooling and resampling with interpolation [2]. In practice, deconvolutional layers are the most commonly used. However, a key problem with the deconvolutional layer is the presence of the checkerboard artifact issue [1, 22], which is caused by the independence among the adjacent pixels on the output feature maps.

To solve the checkerboard artifact problem, some studies focused on improving the post-processing [5], which made the whole process not fully trainable. Some other work focused on adding smooth constraints [13], which resulted in a more complex process. In contrast, the pixel deconvolutional networks [7] solved the checkerboard artifact problem by generating the intermediate feature maps sequentially, thereby building direct relationships among adjacent pixels on output feature maps. This is an effective way while keeping the learning process trainable. However, it only solved this issue in 2D deconvolutional layers. The checkerboard artifact issue also exists in 3D deconvolutional layers, and it is more challenging due to the fact that the number of intermediate maps needed to generate one deconvolutional layer grows exponentially with dimensionality. There are a total of 2^d intermediate feature maps needed when generating a deconvolutional layer given a d dimensional input.

In this paper, we propose the voxel deconvolutional layer (VoxelDCL) to address the checkerboard artifact of 3D deconvolutional layers. We build four variations of VoxelDCLs, which are known as iVoxelDCLc, iVoxelDCLa, VoxelDCLc, and VoxelDCLa, based on different approaches when generating the intermediate feature maps. The iVoxelDCLc and iVoxelDCLa use the input along with the generated intermediate feature maps to generate new intermediate feature maps by concatenation and addition, respectively. The VoxelDCLc and VoxelDCLa only use the generated intermediate feature maps to generate new intermediate feature maps. We also provide an efficient implementation method to improve the computational efficiency by reducing unnecessary dependencies among the intermediate feature maps. To demonstrate the effectiveness of the proposed VoxelDCL, we build the voxel deconvolutional networks (VoxelDCN) based on the U-Net with four variations of VoxelDCL and apply them to the volumetric brain image labeling task [6, 29, 31, 33, 37] using the Alzheimer's disease neuroimaging initiative (ADNI) [20] and the LONI LPBA40 [25] datasets. The results show that our methods outperform the U-Net with regular deconvolutional layers significantly in terms of dice ratio. Specifically, the iVoxelDCLa achieves the best performance on both the ADNI and LONI LPBA40 datasets with 83.34% and 79.12% dice ratio, respectively.

2 RELATED WORK

2.1 Deep Learning for Pixel-Wise Prediction

In computer vision, many problems can be formulated as assigning a label to every pixel of an image, such as semantic segmentation [11, 21, 23, 26]. The goal of semantic segmentation is to label each pixel of the image so that pixels belonging to the same class get the same label. In recent years, convolutional neural networks (CNNs) have had great success in computer vision and brought great improvement in semantic segmentation tasks. The fully convolutional network (FCN) [26] can generate segmentation maps with an input image of any size. It is much faster and more effective than the traditional patch-wise classification methods [3]. Multiple encoder-decoder based architectures [16, 23] are also proposed. The encoder part reduces the spatial sizes with pooling layers and extracts high-level feature maps while the decoder part recovers them to the original spatial size. U-Net [23] is one of the most widely used networks in natural and medical image segmentation tasks.

2.2 Operations for Up-Sampling

To recover the spatial size of the extracted feature maps to the original size as the input image, up-sampling layers are essential in the decoder path of a dense prediction network. Unpooling, resampling with interpolation, and deconvolution are the most commonly used operations for up-sampling. Unpooling layers [21, 34] put each activation back to its original location based on the recorded location of maximum activation selected during the corresponding pooling operations. Up-sampling layers using resampling with interpolation [2] scales an feature map to the desired size and calculates the value of each pixel using an interpolation method such as bilinear interpolation. In the above two methods, there is no learning parameters required in the operation. The third method, known as the deconvolutional layer [1, 22, 28], is the most popular method for up-sampling tasks. It up-samples feature maps by using operations that can be formulated as convolutional operations with learned kernels. More details are explained in Section 3.1. However, deconvolutional layers suffer from the checkerboard artifact issue. This is because there is no direct relationship among adjacent pixels on the output feature map. To solve this problem, pixel deconvolutional networks [7] built direct relationship among adjacent pixels on the output of a deconvolutional layer by generating intermediate feature maps sequentially. This process is different in traditional deconvolutional layers where the intermediate feature maps are generated independently. Details of the concepts of intermediate feature maps and the sequential process are given in Section 3. The dependencies among the intermediate feature maps add direct relationships among adjacent pixels on the output feature map, thereby alleviating the checkerboard artifact in 2D deconvolutional layers.

2.3 3D Brain Image Labeling

3D brain image labeling is an important task since plenty of quantitative brain image analysis often relies on it. It becomes a popular topic in the medical image analysis field. In computer vision area, this task can be seen as a 3D semantic segmentation task [35, 38]. Some existing methods [8, 18, 24, 30, 32, 36] use multi-atlas based labeling models to predict the labels of new images. These methods first transfer the segmentation labels from pre-labeled atlases to a new image and then apply the label fusion method to combine the transferred labels as predicted results. There are also several learning-based labeling methods using random forests, support vector machine, and neural networks [19] to train a prediction model. Among these methods, neural networks have an unique advantage in that it does not require hand-crafted feature extraction in advance. It enables end-to-end learning, which improves efficiency of the training process. In this work, we introduce a novel deep neural network architecture and apply the variations of this network on the 3D brain image labeling task.

3 VOXEL DECONVOLUTIONAL NETWORKS

In this section, we introduce the concept of deconvolutional layers and other related operations, including transposed convolutional layers [28] and sub-pixel convolutional layers [1, 22, 28]. We show the equivalence among these concepts in the 1D case. After that, we describe 2D and 3D deconvolutional layers in the form of sub-pixel deconvolutional layer and show the checkerboard artifact that they suffer [22]. We discuss existing approaches to solve the checkerboard artifact and propose the voxel deconvolutional layers and networks, which can overcome the checkerboard problem of 3D deconvolutional layers. We show that our methods achieve better labeling performance on the Alzheimer's disease Neuroimaging Initiative (ADNI) [20] and the LONI LPBA40 [25] datasets as compared with other baseline methods.

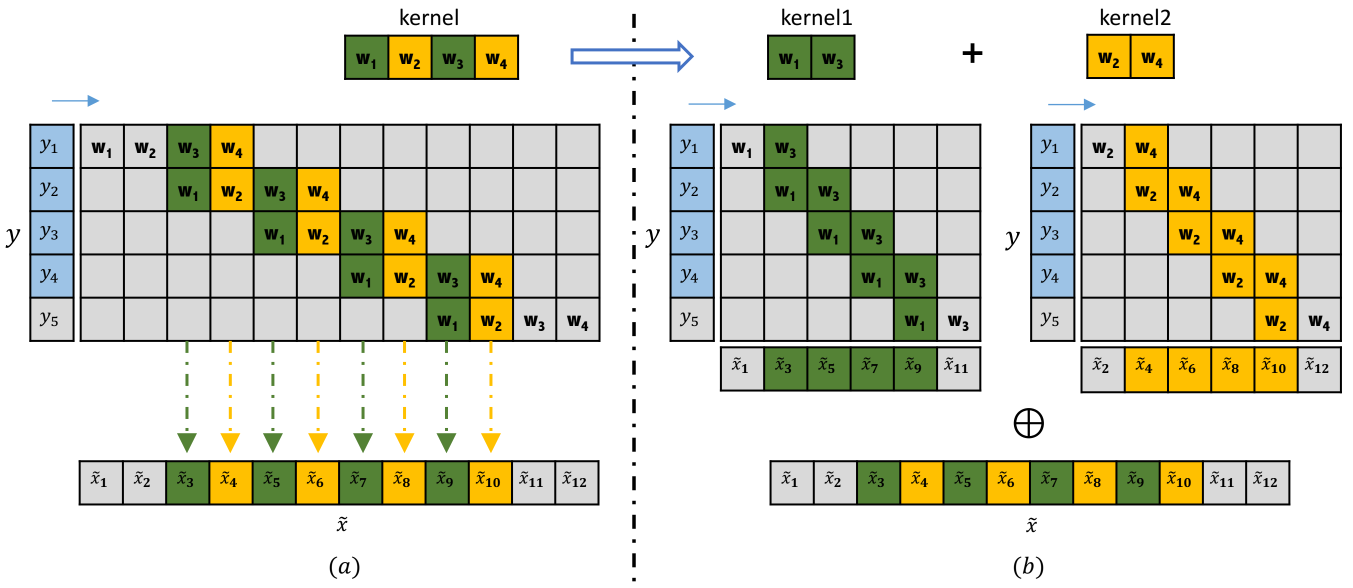

Figure 1: Illustration of a 1D deconvolutional layer. The green and yellow colors represent the odd and even columns of the kernel, respectively. $\oplus$ denotes the periodical shuffling and combination operation. y is the input and $\tilde{x}$ is the output from the deconvolutional layer. By multiplying each column of the weight matrix C with y, we can obtain its corresponding output $\tilde{x}_i$ (see (a)). Since all the odd columns of $\tilde{x}$ are computed only by $w1$ and $w3$ and all the even columns of $\tilde{x}$ are computed by $w2$ and $w4$, the operation can be decomposed into two convolutions with two independent kernels of sizes 1×2, resulting in kernel1 and kernel2. The intermediate results are then shuffled and combined to obtain the final output $\tilde{x}$. $\tilde{x}$ can be cropped by two pixels from both ends to make the up-sampling factor to be 2.

3.1 Deconvolutional Layer

Deconvolutional layers can be viewed as a form of convolutional layers. The relationship between convolutional and deconvolutional layers can be best understood when both of them are considered as fully connected layers. Specifically, convolutional layers can be considered as fully connected layers with sparse connection matrices. Let us consider the following example in 1D. Given an input vector $x \in \mathbb{R}^8$, we pad the input by adding two zeros to both ends of x, yielding the padded input $\tilde{x} \in \mathbb{R}^{12}$. We then apply a 1D convolution with a stride of 2 and a kernel of size 4 to produce an output vector $y \in \mathbb{R}^5$. This convolution operation can be equivalently viewed as a fully connected operation as

$$y = C\tilde{x}, \tag{1}$$

with a 5×12 sparse connection matrix C, defined as follows:

$$C = \begin{bmatrix} w_1 & w_2 & w_3 & w_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & w_1 & w_2 & w_3 & w_4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_1 & w_2 & w_3 & w_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & w_1 & w_2 & w_3 & w_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & w_1 & w_2 & w_3 & w_4 \end{bmatrix}. \tag{2}$$

In this example, y can be cropped by one pixel from the right end to make the down-sampling factor to be 2.

The above example reduces x to a lower-dimension by a convolution of stride 2. A similar operation can be used to convert a lower-dimensional input to a high-dimensional output, resulting in the deconvolutional operation. Specifically, assume y is the input and $\tilde{x}$ is the output, then their relationship can be expressed as

$$\tilde{x} = C^T y, \tag{3}$$

where C is defined in Eq. (2). It can be seen from Eq. (3) that, given a lower-dimensional input, a higher-dimensional output can be obtained via a fully connected layer with a sparse connection matrix. This results in the so-called deconvolutional layer. Since deconvolution layers use the transposed weight matrix of its corresponding convolutional operation C, it is also known as transposed convolutions.

Since C is a sparse matrix, another way to understand the deconvolutional layer is to interpret it as a standard convolution. Specifically, it can be considered as performing multiple independent convolutions on the input followed by a periodical shuffling operation [27]. An example is given in Figure 1 to illustrates this idea.

3.2 3D Deconvolutional Layer

In the previous section, we describe how to understand deconvolutional layers as a form of convolutional layers in 1D space. The same strategy can be used in 2D and 3D spaces as well. In 2D case, the first step is to generate 4 intermediate feature maps from padded input. This step simply requires 2D convolutional operations. Next, these feature maps are periodically shuffled and combined together to form the output. The same steps can be applied to perform deconvolutional operations in 3D space, which will be discussed next.

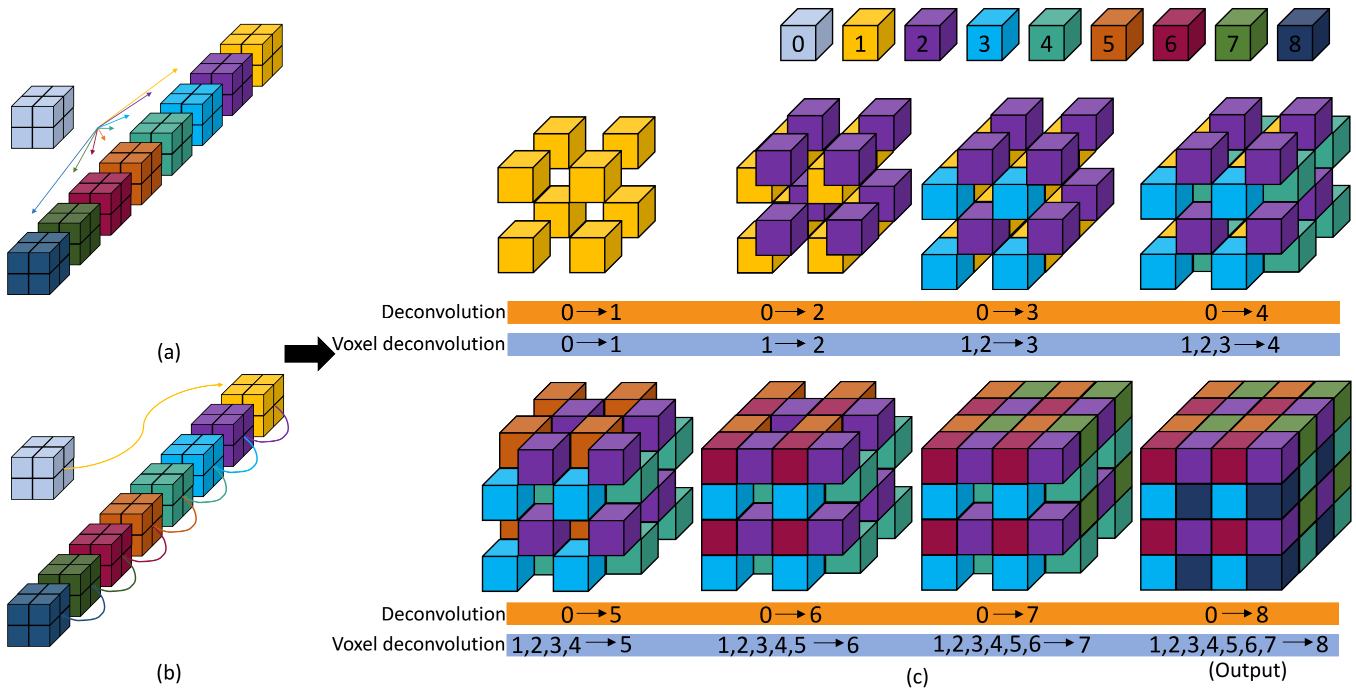

Figure 2: Illustration of the 3D deconvolutional layer and voxel deconvolutional layer. For convenience, we use number 0 to denote the input, and number 1 - 8 to denote the eight intermediate feature maps which are generated from convolutional operations. The input size is $2 \times 2 \times 2$ and the desired out size is $4 \times 4 \times 4$. The size of each intermediate feature map is also $2 \times 2 \times 2$. (a) shows how the intermediate feature maps are generated independently from eight different kernels in a regular 3D deconvolutional layer. This independent generation process is also indicated in the lower orange bar in (c). (b) shows how the intermediate feature maps are generated sequentially in a voxel deconvolutional layer. The i^{th} feature map are built on the $1^{st}, \cdots, (i-1)^{th}$ feature maps. In iVoxelDCLa and iVoxelDCLc layers, input data is also included in this process. This sequential generation process is also indicated in the lower blue bar in (c). (c) displays the periodically shuffling and combination process of eight feature maps to form the final output of a deconvolutional layer.

Assume $X \in \mathbb{R}^{l \times m \times n}$ is the input and $Y \in \mathbb{R}^{2l \times 2m \times 2n}$ is the output of a deconvolutional layer. Here the up-sampling factor is 2. We first perform eight 3D convolutional operations on the input. Each of such operation generates one intermediate feature map Y_i with size $l \times m \times n$ given by

$$Y_i = X \otimes k_i, \quad i = 1, 2, \cdots, 8, \tag{4}$$

where $\otimes$ denotes the convolutional operation, Y_i denotes the i^{th} feature map and k_i denotes the corresponding kernel. We add padding to the input in order to make sure that spatial size remains the same. Finally, these 8 feature maps are periodically shuffled and combined together as

$$Y = Y_1 \oplus Y_2 \oplus Y_3 \oplus Y_4 \oplus Y_5 \oplus Y_6 \oplus Y_7 \oplus Y_8, \tag{5}$$

where $\oplus$ denotes the periodical shuffling and combination operation, as illustrated in Figure 2.

Compared to deconvolutional layers in 2D space, the number of feature maps needed in 3D space grows exponentially. In general, assume the up-sampling factor is 2 and the dimensionality is d, the number of intermediate feature maps need to generate one deconvolutional layer is 2^d. This indicates that deconvolutional layers in higher dimensional space are much more complex and computationally expensive than those in lower dimensional space. These are factors to consider when we introduce voxel deconvolutional layers in Section 3.3.

3.3 Voxel Deconvolutional Layer

When we consider a deconvolutional layer as the shuffled combination of several intermediate feature maps, discontinuities among adjacent pixels can be observed frequently. This is not surprising since the adjacent pixels are coming from independent feature maps. This is the checkerboard artifact problem that all deconvolutional layers suffer, no matter calculated in 2D or 3D space.

Gao *et al.* [7] solved the checkerboard artifact of deconvolutional layers in 2D space by adding relations to the intermediate feature maps. Inspired by [7], we propose the voxel deconvolutional layer to solve the same issue in 3D space. As we mentioned in Section 3.2, the computational cost and complexity of deconvolutional operations in 3D space are much higher than those in 2D space, thereby making this task challenging.

Back to the example in Section 3.2, we use $X \in \mathbb{R}^{l \times m \times n}$ to denote the input and $Y \in \mathbb{R}^{2l \times 2m \times 2n}$ to denote the output. The eight intermediate feature maps are $Y_1, Y_2, \cdots, Y_8$, each of size $l \times m \times n$.

In voxel deconvolutional layers, $Y_1, Y_2, \cdots, Y_8$ are formed sequentially to build direct relations in-between. Suppose $Y_1, Y_2, \cdots, Y_{i-1}$ are already generated, Y_i will be built on X together with $Y_1, Y_2, \cdots, Y_{i-1}$ instead of simple X as in traditional deconvolutional layers. The generation of Y_1 remains the same as it is the first intermediate feature map. There are mainly two ways to combine $X, Y_1, Y_2, \cdots, Y_{i-1}$ together; namely addition or concatenation.

We first introduce the naming conventions of the proposed voxel deconvolutional layers and corresponding networks in below:

- **Suffix c:** We use concatenation when combining multiple inputs to generate a new intermediate feature map.
- **Suffix a:** We use addition when combining multiple inputs to generate a new intermediate feature map.
- **Prefix iVoxel:** Both the original input and existing intermediate feature maps are included to generate a new feature map.
- **Prefix Voxel:** Only existing intermediate feature maps are included to generate a new feature map.

We first introduce **iVoxelDCLc**. In iVoxelDCLc, the input X and intermediate feature maps $Y_1, Y_2, \cdots, Y_{i-1}$ are concatenated together to form the input to generate Y_i as

$$Y_1 = X \otimes k_1; Y_i = [X, Y_1, , \cdots, Y_{i-1}] \otimes k_i, \text{ for } i = 2, \cdots, 8, \quad (6)$$

where $\otimes$ denotes a convolutional operation, $[\cdot, \cdot]$ represents the concatenation and k_i denotes the corresponding convolutional kernel. By using concatenation, $X, Y_1, \cdots, Y_{i-1}$ are stacked as channels of a single input to generate Y_i. There will be weights assigned to each of $X, Y_1, \cdots, Y_{i-1}$, indicating their importance to Y_i. However, concatenation will cause memory and computational issues due to the large amount of parameters needed. This problem can be reduced by using addition when combining multiple related feature maps as input, which will be denoted as iVoxelDCLa.

In **iVoxelDCLa**, the input X and intermediate feature maps $Y_1, Y_2, \cdots, Y_{i-1}$ are added together to form the input to generate Y_i as

$$Y_1 = X \otimes k_1; Y_i = \left(X + \sum_{j=1}^{i-1} Y_j \right) \otimes k_i, \text{ for } i = 2, \cdots, 8. \quad (7)$$

In iVoxelDCLc and iVoxelDCLa, the input X is included in the generation of every intermediate feature map. This might be redundant since the information carried in X will be carried over in Y_1 to Y_8 through this sequential process. In the next two versions of VoxelDCL, we take out the input X when generating $Y_2, \cdots, Y_8$. Only the first intermediate feature map Y_1 is generated using X. Similarly, we first use concatenation to combine multiple inputs, which will be denoted as VoxelDCLc.

In **VoxelDCLc**, the intermediate feature maps $Y_1, Y_2, \cdots, Y_{i-1}$ are concatenated together to form the input to generate Y_i while Y_1 is generated from the input X:

$$Y_1 = X \otimes k_1; Y_i = [Y_1, \cdots, Y_{i-1}] \otimes k_i, \text{ for } i = 2, \cdots, 8. \quad (8)$$

To further reduce memory usage and computational complexity, we use addition instead of concatenation the same way as we described in iVoxelDCLa layer. This layer is known as VoxelDCLa.

In **VoxelDCLa**, intermediate feature maps $Y_1, Y_2, \cdots, Y_{i-1}$ are added together to form the input to generate Y_i while Y_1 is generated from the input X:

$$Y_1 = X \otimes k_1; Y_i = \left(\sum_{j=1}^{i-1} Y_j \right) \otimes k_i, \text{ for } i = 2, \cdots, 8. \quad (9)$$

Once all the eight intermediate feature maps are generated, we can obtain the final output using Eq. (5). In Figure 2, we provide an example to illustrate how the VoxelDCLa and VoxelDCLc work.

3.4 Efficient Implementation

We propose an efficient way to implement all of the above voxel deconvolutional layers in order to scale down the computational cost. The goal is to reduce unnecessary dependencies among the intermediate feature maps. The basic principle is that, for voxels that are not adjacent to each other but linked to the same pivot voxel in the output, we generate their corresponding feature maps in parallel. In this way, we remove the dependency of unrelated inputs when generating new feature maps, hence simplifying the process and expediting the computation. In this new design, the intermediate feature maps are generated in 4 steps. Specifically, Y_1, which denotes the 1^{st} intermediate feature map, is generated from the input X. Then Y_2 is generated from Y_1. Since voxels from Y_3, Y_4, Y_5 are not adjacent to each other but are all linked to voxels from Y_1, we generate them in parallel using previously generated Y_1 and Y_2. The same strategy is used for generating Y_6, Y_7, Y_8. They are generated in parallel from all the previously generated $Y_1, \cdots, Y_5$. In iVoxelDCLa and iVoxelDCLc, the input X is also included in each step. By using parallel generation in steps 3 and 4, the computational time for VoxelDCL is reduced. In this way, we build a reasonable relationship among these intermediate feature maps. Figure 3 illustrates how VoxelDCLa and VoxelDCLc work.

3.5 Overview of Network Architectures

U-Net [23] is a convolutional neural network architecture commonly used for dense prediction. It has an encoder path which extracts high-level feature maps from raw images and a decoder path which recovers feature maps to the original spatial size. In the classic U-Net architecture, there is repeated application of three convolutional layers, followed by a max pooling layer for down-sampling in each encoder block. Batch normalization [10] is used after each convolutional layer. The number of channels is doubled after each down-sampling operation. In each decoder block, there is a deconvolutional layer followed by two convolutional layers. The output of the deconvolutional layers are concatenated with corresponding feature maps from encoder blocks before going to the next block. The proposed architecture in this work, which is denoted as voxel deconvolutional network (VoxelDCN), is based on this U-Net framework. But we replace the deconvolutional layers in the decoder path with voxel deconvolutional layers. So there are five architectures implemented for comparison:

- **U-Net:** We use U-Net as our baseline method. It is a classic neural network architecture with encoder and decoder parts. It uses deconvolutional layers for up-sampling in decoder path.
- **iVoxelDCNc:** In this method, we replace all deconvolutional layers in U-Net with iVoxelDCLc layers.

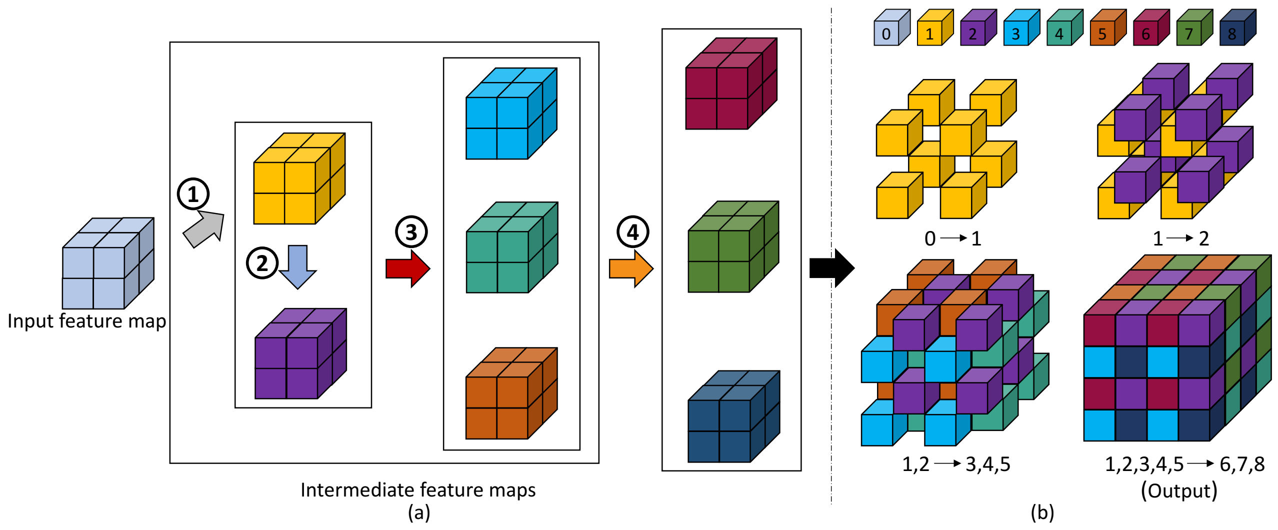

Figure 3: Illustration of the proposed efficient implementation of the VoxelDCLa and VoxelDCLc layers. The notation remains the same as introduced in Figure 2. In addition, each step in the efficient implementation is denoted as numbers with circles in (a). (a) shows the efficient generation process of intermediate feature maps step by step. Step 1 is to generate the 1^{st} feature map directly from the input. Step 2 is to generate the 2^{nd} feature map from the 1^{st} one. The 3^{rd} to 5^{th} feature maps are generated from the 1^{st} and 2^{nd} feature maps in step 3. Finally, the last three feature maps are generated from all previous five feature maps in step 4. (b) shows how these eight intermediate feature maps are shuffled and combined to form the final $4 \times 4 \times 4$ output. In VoxelDCLa layers, multiple feature maps are added together to form the input for the next feature map while in VoxelDCLc layers they are concatenated.

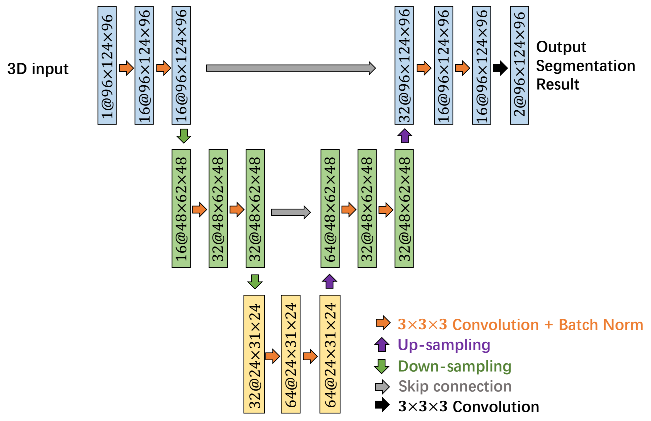

Figure 4: Network architecture implemented on the ADNI dataset. The encoder has two max pooling layers with sizes of 2×2×2. We replace the regular deconvolutional layers in U-Net with variations of the proposed voxel deconvolutional layers in the decoder path.

- **iVoxelDCNa:** In this method, we replace all deconvolutional layers in U-Net with iVoxelDCLa layers.
- **VoxelDCNc:** In this method, we replace all deconvolutional layers in U-Net with VoxelDCLc layers.
- **VoxelDCNa:** In this method, we replace all deconvolutional layers in U-Net with VoxelDCLa layers.

Figure 4 shows the architecture we implemented on the ADNI dataset. On the LONI LPBA40 dataset, we use a similar architecture with more blocks in both encoder and decoder paths to avoid underfitting problem.

4 EXPERIMENTAL STUDIES

In this section, we demonstrate the performance of our proposed methods on two public datasets that have been widely used for brain images labeling, namely the Alzheimer's disease neuroimaging initiative (ADNI) dataset [20] and the LONI LPBA40 dataset [25]. We select these two datasets since they represent different types of brain image labeling. The ADNI dataset has rich brain MR images for labeling hippocampal regions. It can be treated as a segmentation task with binary classes. On the contrary, the LONI LPBA40 dataset has various regions of interest, mostly inside the cortex of the brain. It can be treated as a multi-class segmentation task. Examples from these datasets are provided in Figure 5. Our code is publicly available[1].

We use classic U-Net as the baseline method to evaluate the performance of our proposed methods . Dice ratio is used to measure the labeling accuracy. It calculates the degree of overlap between the predicted area and its corresponding ground truth for each target class. It is defined by

$$DICE(A, B) = \frac{2|A \cap B|}{|A| + |B|}, \tag{10}$$

[1]https://github.com/divelab/VoxelDCN

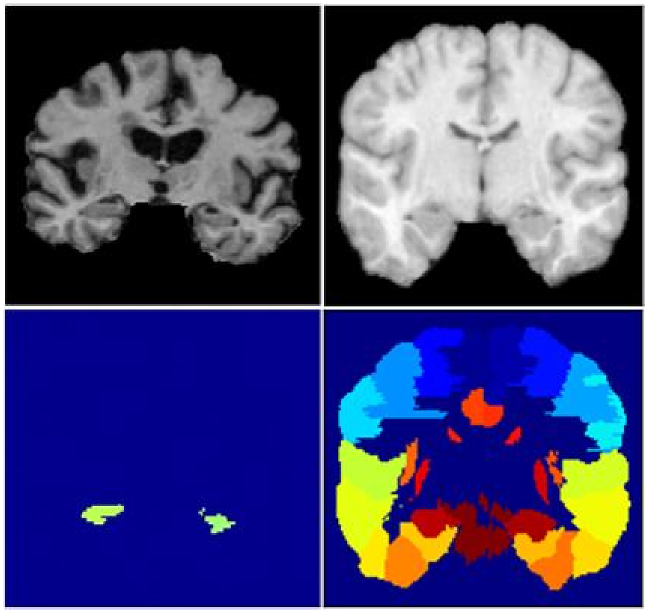

Figure 5: Examples of MR images from the ADNI dataset (left) and the LONI LPBA40 dataset (right). The top row displays the intensity images, and the bottom row displays the manually segmented labels.

where A is the predicted result and B is the corresponding true label. For k different regions of interest (ROI), an averaged dice ratio is calculated, given by

$$DICE = \frac{1}{k}\sum_{i=1}^{k} DICE(A_i, B_i) = \frac{1}{k}\sum_{i=1}^{k}\frac{2|A_i \cap B_i|}{|A_i| + |B_i|}, \quad (11)$$

where A_i represents the prediction for the i^{th} ROI and B_i represents the corresponding ground truth label.

4.1 ADNI Dataset

The ADNI dataset is mostly used to label hippocampus in brains. The task is to determine whether a certain part of the brain is hippocampus or not. We consider it as a segmentation task with binary classes.

Data Preparation: There are 64 samples in the ADNI dataset with sizes of $96 \times 124 \times 96$. We randomly split the dataset into training data with 59 samples and testing data with 5 samples. We repeat this process for five times and build five segmentation models respectively. The averaged dice ratio is the reported performance of this experiment.

Experimental Setup: We build the baseline model using U-Net architecture as shown in Figure 4. The number of output channels in the first encoder block is 16. The convolutional kernel size is $3\times3\times3$ and the stride size used in the max pooling layers is $2\times2\times2$. The last convolutional operation in the final decoder block has 2 output channels, which corresponds to the total number of classes. We apply batch normalization after each convolutional layer, except for the output layer. The training batch size is 4. We implement our methods on Tensorflow and use AdamOptimizer to update the network with moment estimates $\beta1 = 0.9$ and $\beta2 = 0.999$.

Table 1: Experimental results on the ADNI and LONI LPBA40 datasets.

Dataset	Model	Dice Ratio (%)
ADNI	U-Net	82.1985
	iVoxelDCNc	82.8784
	iVoxelDCNa	**83.3403**
	VoxelDCNc	82.2777
	VoxelDCNa	82.9553
LONI LPBA40	U-Net	77.4209
	iVoxelDCNc	78.3524
	iVoxelDCNa	**79.1294**
	VoxelDCNc	78.7590
	VoxelDCNa	77.7654

Analysis of Results: Table 1 shows the overall dice ratios of the baseline method and the proposed methods. We can see that all the proposed methods have higher dice ratios than the baseline method. Model iVoxelDCNa achieves 83.34% in dice ratio, which improves the baseline method by 1.39%. The result demonstrates that the proposed methods, which intend to build relationships among intermediate feature maps, can help networks better capture local information of images than the baseline method.

4.2 LONI LPBA40 Dataset

The LONI LPBA40 dataset is designed as a multi-class labeling task. It requires 54 regions of interest to be labeled in brain images automatically.

Data Preparation There are 40 brain image samples with sizes of $220 \times 220 \times 220$ in the dataset. Each image has 54 manually labeled ROIs along with cerebrum, brainstem, and background. When calculating result, only dice ratios of the 54 ROIs are included. Since we are not interested in predicting background, we first crop the image to drop the excessive background area around boundaries. The cropped image size is $180\times145\times137$. Then we split the dataset into training set with 16 samples, validation set with 4 samples and testing set with the remaining 20 samples. We flip the images across all three directions on training and validation datasets. This increases the number of training samples to 128 and the number of validation samples to 32. The size of testing data remains 20 as we do not apply data augmentation on testing data.

Experimental Setup: The architecture of U-Net implemented on this dataset consists of five blocks in both the encoder path and decoder path. The number of output channels in the first encoder block is 32. The convolutional kernel size is $3\times3\times3$ and stride size used in the max pooling layers is $1\times2\times2$. The last convolutional operation in the final decoder block has 57 output channels, which corresponds to the total number of classes (54 ROIs, cerebrum, brainstem and background). The training batch size is 1. Since each data sample provided is too large to be stored in memory, we use training patch with size of $128\times128\times32$ as input in the training process. Each training patch are randomly cropped from training data. At the testing time, we pick one patch at a time from a single testing data sample with the same size in turn. The predicted patches

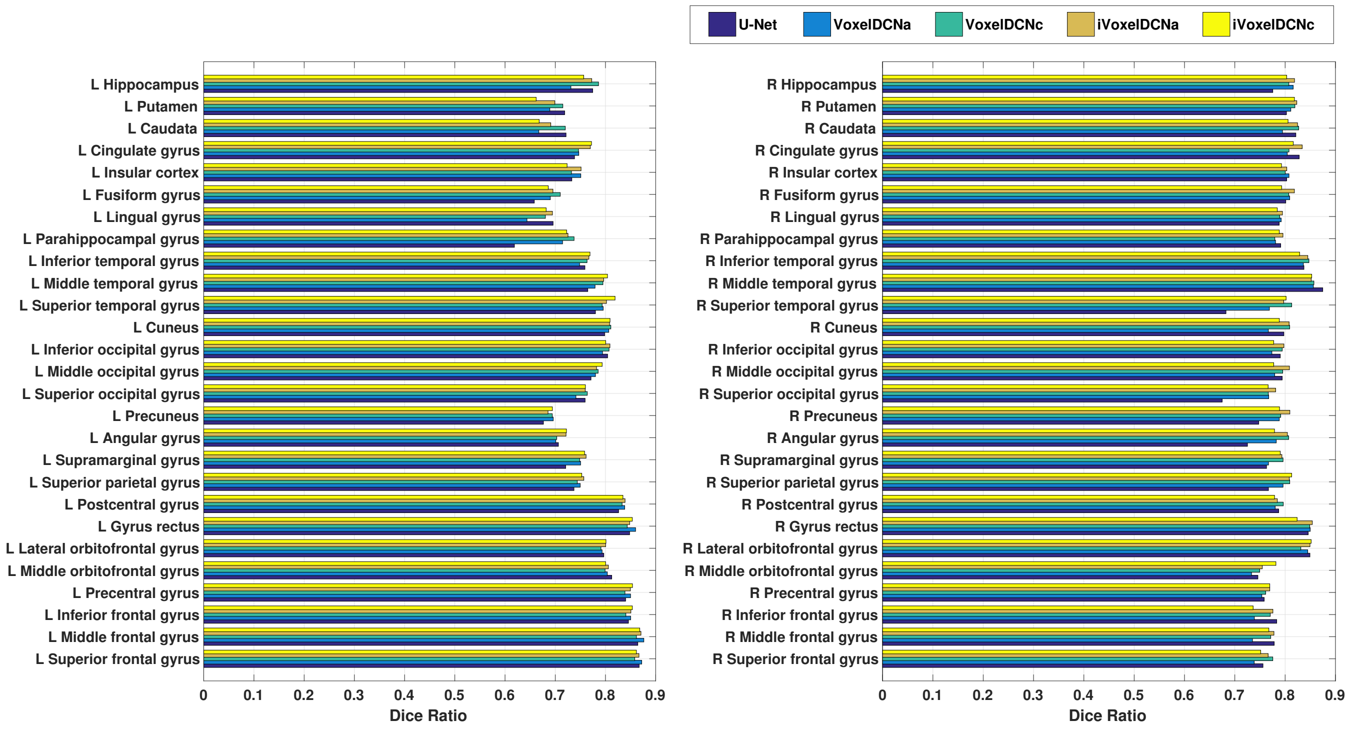

Figure 6: Experimental results on the LONI LPBA40 datasets. Each set of bars represents the dice ratios of 3D U-Net, VoxelDCLa, VoxelDCLc, iVoxelDCLa, iVoxelDCLc on testing data for each ROI.

are concatenated together with a linear weighted summation on overlapped part to get the final prediction results. We implement our models on Tensorflow and use AdamOptimizer to update the network with moment estimates $\beta 1 = 0.9$ and $\beta 2 = 0.999$.

Analysis of Results: The dice ratio of ROIs in the left and the right hemispheres are provided in Figure 6. Compared with the baseline method, iVoxelDCNa shows improvement in 44 out of 54 ROIs; VoxelDCNc shows improvement in 35 out of 54 ROIs; iVoxelDCNc shows improvement in 30 out of 54 ROIs; VoxelDCNa shows improvement in 29 of 54 ROIs. Table 1 shows the experimental results in terms of average dice ratio. All the proposed methods have better performance than the baseline model. Network iVoxelDCNa has the best performance and achieves 2.21% improvement compared with baseline. The result demonstrates that the proposed methods can help networks better capture the local information of images than the baseline method by eliminating the checkerboard artifact, yielding a better labeling result.

4.3 Timing Comparison

Table 2 shows the comparison of the training and testing time between baseline and proposed methods. We can see that iVoxelDCLa and iVoxelDCLc take more time to train than VoxelDCLa and VoxelDCLc in that they take original input when generating each intermediate feature map. In VoxelDCL, the use of concatenation slightly increases the training and testing time. All the proposed methods take more time to train and test than the baseline model, but the increase is not dramatic. Overall, we do not expect this to be a major bottleneck of the proposed methods.

5 CONCLUSION AND DISCUSSION

In this work, we propose the VoxelDCL to address the checkerboard artifact problem of 3D deconvolutional layers. VoxelDCL generates the intermediate feature maps of 3D deconvolutional layers sequentially, thereby building relationships among the adjacent voxels on the output feature maps. We then build four variations of the voxel deconvolutional networks (VoxelDCN) and apply them to the ADNI and LONI PBA40 datasets for volumetric brain image labeling tasks. Experimental results demonstrate the effectiveness of our methods. All of the proposed methods outperform U-Net with regular 3D deconvolutional layers. This indicates that developing dependencies among the intermediate feature maps in deconvolutional layers indeed alleviates the checkerboard artifact issue, thereby improving the prediction results. In out current study, we apply the VoxelDCL to U-Net for voxel-wise prediction tasks. Generally, this layer can be applied to various 3D deep network architectures with up-sampling operations. We plan to apply the proposed methods to models like 3D generative adversarial networks for image generation task in the future.

ACKNOWLEDGMENTS

This work was supported in part by National Science Foundation grants DBI-1641223, DBI-1661289, IIS-1615035, and National Institutes of Health grants EB008374 and AG041721.

Table 2: Training and testing time on the two datasets using Tesla K80 GPU. We compare the training time of 10,000 (ADNI) and 100,000 iterations (LONI LPBA40), and testing time of 5 (ADNI) and 20 testing subjects for the baseline U-Net and the proposed methods.

Dataset	Model	Training time	Testing time
ADNI	U-Net	45 h 43 min	19.45 sec
	VoxelDCNa	54 h 57 min	20.80 sec
	VoxelDCNc	57 h 12 min	19.18 sec
	iVoxelDCNa	59 h 31 min	19.87 sec
	iVoxelDCNc	59 h 54 min	19.65 sec
LONI LPBA40	U-Net	52 h 05 min	12 min 12 sec
	VoxelDCNa	60 h 37 min	20 min 35 sec
	VoxelDCNc	63 h 24 min	19 min 85 sec
	iVoxelDCNa	71 h 12 min	20 min 32 sec
	iVoxelDCNc	75 h 40 min	20 min 05 sec

REFERENCES

[1] Andrew Aitken, Christian Ledig, Lucas Theis, Jose Caballero, Zehan Wang, and Wenzhe Shi. 2017. Checkerboard artifact free sub-pixel convolution: A note on sub-pixel convolution, resize convolution and convolution resize. *arXiv preprint arXiv:1707.02937* (2017).

[2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. 2016. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915* (2016).

[3] Dan Ciresan, Alessandro Giusti, Luca M Gambardella, and Jürgen Schmidhuber. 2012. Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in neural information processing systems.* 2843–2851.

[4] Angela Dai, Daniel Ritchie, Martin Bokeloh, Scott Reed, Jürgen Sturm, and Matthias Nießner. 2017. ScanComplete: Large-Scale Scene Completion and Semantic Segmentation for 3D Scans. *arXiv preprint arXiv:1712.10215* (2017).

[5] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. 2015. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision.* 2758–2766.

[6] Christine Fennema-Notestine, Donald J Hagler, Linda K McEvoy, Adam S Fleisher, Elaine H Wu, David S Karow, and Anders M Dale. 2009. Structural MRI biomarkers for preclinical and mild Alzheimer's disease. *Human brain mapping* 30, 10 (2009), 3238–3253.

[7] Hongyang Gao, Hao Yuan, Zhengyang Wang, and Shuiwang Ji. 2017. Pixel Deconvolutional Networks. *arXiv preprint arXiv:1705.06820* (2017).

[8] Yaozong Gao, Shu Liao, and Dinggang Shen. 2012. Prostate segmentation by sparse representation based classification. *Medical physics* 39, 10 (2012), 6372–6387.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems.* 2672–2680.

[10] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning.* 448–456.

[11] Md Amirul Islam, Neil Bruce, and Yang Wang. 2016. Dense Image Labeling Using Deep Convolutional Neural Networks. In *Computer and Robot Vision (CRV), 2016 13th Conference on.* IEEE, 16–23.

[12] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 2013. 3D Convolutional Neural Networks for Human Action Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 1 (2013), 221–231.

[13] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision.* Springer, 694–711.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems.* 1097–1105.

[15] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[16] Kisuk Lee, Jonathan Zung, Peter Li, Viren Jain, and H Sebastian Seung. 2017. Superhuman Accuracy on the SNEMI3D Connectomics Challenge. *arXiv preprint arXiv:1706.00120* (2017).

[17] Rongjian Li, Wenlu Zhang, Heung-Il Suk, Li Wang, Jiang Li, Dinggang Shen, and Shuiwang Ji. 2014. Deep Learning Based Imaging Data Completion for Improved Brain Disease Diagnosis. In *Proceedings of the 17th International Conference on Medical Image Computing and Computer Assisted Intervention.* 305–312.

[18] Guangkai Ma, Yaozong Gao, Guorong Wu, Ligang Wu, and Dinggang Shen. 2016. Nonlocal atlas-guided multi-channel forest learning for human brain labeling. *Medical physics* 43, 2 (2016), 1003–1019.

[19] Vincent A Magnotta, Dan Heckel, Nancy C Andreasen, Ted Cizadlo, Patricia Westmoreland Corson, James C Ehrhardt, and William TC Yuh. 1999. Measurement of brain structures with artificial neural networks: two-and three-dimensional applications. *Radiology* 211, 3 (1999), 781–790.

[20] Susanne G Mueller, Michael W Weiner, Leon J Thal, Ronald C Petersen, Clifford Jack, William Jagust, John Q Trojanowski, Arthur W Toga, and Laurel Beckett. 2005. The Alzheimer's disease neuroimaging initiative. *Neuroimaging Clinics of North America* 15, 4 (2005), 869–877.

[21] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. 2015. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision.* 1520–1528.

[22] Augustus Odena, Vincent Dumoulin, and Chris Olah. 2016. Deconvolution and Checkerboard Artifacts. *Distill* (2016). https://doi.org/10.23915/distill.00003

[23] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention.* Springer, 234–241.

[24] Gerard Sanroma, Guorong Wu, Yaozong Gao, and Dinggang Shen. 2014. Learning to rank atlases for multiple-atlas segmentation. *IEEE transactions on medical imaging* 33, 10 (2014), 1939–1953.

[25] David W Shattuck, Mubeena Mirza, Vitria Adisetiyo, Cornelius Hojatkashani, Georges Salamon, Katherine L Narr, Russell A Poldrack, Robert M Bilder, and Arthur W Toga. 2008. Construction of a 3D probabilistic atlas of human cortical structures. *Neuroimage* 39, 3 (2008), 1064–1080.

[26] Evan Shelhamer, Jonathan Long, and Trevor Darrell. 2017. Fully convolutional networks for semantic segmentation. *IEEE transactions on pattern analysis and machine intelligence* 39, 4 (2017), 640–651.

[27] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. 2016. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 1874–1883.

[28] Wenzhe Shi, Jose Caballero, Lucas Theis, Ferenc Huszar, Andrew Aitken, Christian Ledig, and Zehan Wang. 2016. Is the deconvolution layer the same as a convolutional layer? *arXiv preprint arXiv:1609.07009* (2016).

[29] Tong Tong, Robin Wolz, Joseph V Hajnal, and Daniel Rueckert. 2012. Segmentation of brain MR images via sparse patch representation. In *MICCAI Workshop on Sparsity Techniques in Medical Imaging (STMI).*

[30] Hongzhi Wang, Jung W Suh, Sandhitsu R Das, John B Pluta, Caryne Craige, and Paul A Yushkevich. 2013. Multi-atlas segmentation with joint label fusion. *IEEE transactions on pattern analysis and machine intelligence* 35, 3 (2013), 611–623.

[31] Li Wang, Yaozong Gao, Feng Shi, Gang Li, John H Gilmore, Weili Lin, and Dinggang Shen. 2015. LINKS: Learning-based multi-source IntegratioN frameworK for Segmentation of infant brain images. *NeuroImage* 108 (2015), 160–172.

[32] Guorong Wu, Minjeong Kim, Gerard Sanroma, Qian Wang, Brent C Munsell, Dinggang Shen, Alzheimer's Disease Neuroimaging Initiative, et al. 2015. Hierarchical multi-atlas label fusion with multi-scale feature representation and label-specific patch partition. *NeuroImage* 106 (2015), 34–46.

[33] Guorong Wu, Qian Wang, Daoqiang Zhang, Feiping Nie, Heng Huang, and Dinggang Shen. 2014. A generative probability model of joint label fusion for multi-atlas based brain segmentation. *Medical image analysis* 18, 6 (2014), 881–890.

[34] Matthew D Zeiler, Graham W Taylor, and Rob Fergus. 2011. Adaptive deconvolutional networks for mid and high level feature learning. In *Computer Vision (ICCV), 2011 IEEE International Conference on.* IEEE, 2018–2025.

[35] Tao Zeng, Bian Wu, and Shuiwang Ji. 2017. DeepEM3D: Approaching human-level performance on 3D anisotropic EM image segmentation. *Bioinformatics* 33, 16 (2017), 2555–2562.

[36] Lichi Zhang, Qian Wang, Yaozong Gao, Guorong Wu, and Dinggang Shen. 2016. Automatic labeling of MR brain images by hierarchical learning of atlas forests. *Medical physics* 43, 3 (2016), 1175–1186.

[37] Shaoting Zhang, Yiqiang Zhan, Maneesh Dewan, Junzhou Huang, Dimitris N Metaxas, and Xiang Sean Zhou. 2012. Towards robust and effective shape modeling: Sparse shape composition. *Medical image analysis* 16, 1 (2012), 265–277.

[38] Wenlu Zhang, Rongjian Li, Houtao Deng, Li Wang, Weili Lin, Shuiwang Ji, and Dinggang Shen. 2015. Deep convolutional neural networks for multi-modality isointense infant brain image segmentation. *NeuroImage* 108 (2015), 214–224.