Deep Learning Segmentation of Optical Microscopy Images Improves 3D Neuron Reconstruction

Rongjian Li*, Tao Zeng*, Hanchuan Peng, and Shuiwang Ji, *Senior Member, IEEE*

Abstract—Digital reconstruction, or tracing, of 3-dimensional (3D) neuron structure from microscopy images is a critical step toward reversing engineering the wiring and anatomy of a brain. Despite a number of prior attempts, this task remains very challenging, especially when images are contaminated by noises or have discontinued segments of neurite patterns. An approach for addressing such problems is to identify the locations of neuronal voxels using image segmentation methods prior to applying tracing or reconstruction techniques. This preprocessing step is expected to remove noises in the data, thereby leading to improved reconstruction results. In this work, we proposed to use 3D Convolutional neural networks (CNNs) for segmenting the neuronal microscopy images. Specifically, we designed a novel CNN architecture that takes volumetric images as the inputs and their voxel-wise segmentation maps as the outputs. The developed architecture allows us to train and predict using large microscopy images in an end-to-end manner. We evaluated the performance of our model on a variety of challenging 3D microscopy images from different organisms. Results showed that the proposed methods improved the tracing performance significantly when combined with different reconstruction algorithms.

Index Terms—Deep learning, image denoising, image segmentation, neuron reconstruction, BigNeuron.

1 Introduction

Morphology of neurons plays a critical role in the function of the brain. In the nervous system, the electrical impulses and chemical materials are transported through pathways between connected neurons. The study of 3D morphology of individual neurons allows for a precise identification of neuronal pathways and functions, including the neuron phenotype, connectivity, synaptic integration, firing properties, and ultimately the role in neural circuits. Therefore, characterizing the 3D morphology of individual neurons is fundamentally important for understanding the organization of neurons and, furthermore, the mechanism of the nervous system. Recently, many efforts have been devoted to develop automatic or semi-automatic neuron reconstruction algorithms based on microscopy images. However, this reconstruction task remains very challenging when a 3D microscopy image has low signal-to-noise ratio (SNR) and discontinued segments of neurite patterns. Indeed, microscopy image sets with low SNR are quite common for the nervous systems of different animals [1], [2], [3], [4]. Therefore, an effective and automatic denoising algorithm for these challenging situations would substantially amplify the impact of neuron morphology reconstruction.

Currently, most of prior methods for neuronal structure reconstruction on noisy images have used the raw images in an unsupervised way [5], [6], [7], [8]. [1] developed a graph-augmented deformable model to reconstruct the 3D structure of a neuron when the neuron contains a broken structure and/or fuzzy boundary. However, this method still has difficulties on some noisy patterns, *e.g.*, when two parallel tracts are very close and one is brighter than the other. In [1], [9], [10], fast-marching based methods were combined with different pruning techniques to handle 3D noisy images. However, these methods either have low accuracy when the images are anisotropic or are time-consuming when the number of noisy voxels is large. The recently published work [11] developed an automatic tracing framework with training steps to refine the reconstruction results generated by an initial tracing algorithm. One main bottleneck of this method is its relatively high computational complexity.

In this work, we proposed to use the convolutional neural networks (CNNs) for improving the reconstruction performance of existing methods on noise-contaminated images over different organisms. CNNs are a type of fully trainable models that learn a hierarchy of features through nonlinear mappings between multiple stacked layers. CNNs have been widely used in a number of applications and achieved state-of-the-art performance on tasks including large-scale image and video recognition [12], [13], [14], [15], digit recognition [16], and object recognition tasks [17]. Recently, many attempts have been made to extend these models to the field of image segmentation, leading to improved performance [18], [19], [20], [21], [22], [23]. One appealing property of CNNs is that the learned features through trainable parameters can capture highly nonlinear relationship between inputs and outputs. Therefore, the reconstruction methods based on CNNs can be broadly

*These authors contributed equally to this work.

- *R. Li was with the Department of Computer Science, Old Dominion University, Norfolk, VA 23529.*
- *T. Zeng and S. Ji are with the School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164. E-mails: {tzeng,sji}@eecs.wsu.edu*
- *H. Peng is with the Allen Institute for Brain Science, Seattle, WA 98109. E-mail: hanchuanp@alleninstitute.org*

Manuscript received ; revised .

applied to a variety of different data sets.

Specifically, we built a voxel CNN classifier that predicts the probability of every individual voxel in a given image being a part of neuron or not. CNNs have been primarily applied on 2D images in the prior studies. To effectively incorporate the 3D spatial information in neuronal structures, we propose to perform 3D convolution in the convolutional layers of CNNs so that discriminative features along three spatial dimensions are all captured. Considering the large varieties of neuron morphologies among samples, we adopt residual learning and inception learning to build the deep network containing multi-scale information of neural structures. We also use fully convolutional network for increasing the application flexibility of proposed network on different input sizes. Our CNN classifier accepts raw voxel intensities as input without any preprocessing and learns highly discriminative features automatically for producing final probability maps. These probability maps were integrated later with the raw intensities to produce the final adjusted image as input for tracing algorithms. In addition, to obtain the predictions of CNNs on test images, the conventional approaches used patch-based predictions. This approach resulted in significant redundancy in computation, thus making the prediction of large images very time-consuming. To reduce the computational complexity, we applied a stack of deconvolution layers in the CNN architecture that can produce a dense pixel-wise prediction very efficiently. We compared the performance of our approach with that of commonly used tracing methods on a number of challenging 3D neuronal images from different model organisms. Results showed that the proposed deep learning model could enhance the input to existing tracing methods, which thus significantly improved the neural reconstruction performance of prior methods.

2 Material and methods

2.1 BigNeuron atlas

The BigNeuron [24] is a community-contributed project for defining and advancing the state-of-the-art of single neuron reconstruction. It was motivated by a number of recent progresses in neuroinformatics field such as the worldwide neuron reconstruction contest named DIADEM [25], [26]. Currently, BigNeuron provides a large community-oriented neuron morphology database and reconstruction algorithms contributed by many research labs worldwide. There are about 20,000 volumetric optical microscopy images from a variety of species including fruit fly, fish, turtle, chicken, mouse, rat, and human. The neurons in these images are mainly located in regions such as cortical and subcortical areas, retina, and peripheral nervous system. Each image in BigNeuron has single color channel, and contains a single neuron or disconnected multiple neurons having relatively clear separation in their arborization patterns. Some images have the corresponding reconstructions manually curated and/or proofread to be used as references or 'gold standards' for evaluating the automatically produced reconstructions. BigNeuron also provides an open-source cross-platform package 'Vaa3D' [27] for facilitating the bench-testing of neuron tracing algorithms developed by worldwide researchers. A total of 23 neuron-tracing algorithms for neuron quantifications have already been ported to Vaa3D as plug-ins including manual, semi-automatic, and completely automated digital tracing.

2.2 Overview of CNNs

Convolutional neural networks (CNNs) are a class of deep learning models that attempt to compute high-level representations of data using multiple layers of nonlinear transformations. CNNs mimic the visual information processing in the vision system of the brain by applying local filters to the input. Such filters can be trained to extract various local features. To generate specific features of the entire visual field, a sliding filter is applied across entire visual field. Such feature extraction method is also known as parameter sharing and leads to dramatic reduction in the number of trainable parameters. CNN models usually consist of alternating combination of convolutional layers and local neighborhood pooling layers, resulting in complex hierarchical representations of the inputs. These properties make CNN a powerful tool in image-related applications. Prior studies have shown that CNNs achieved superior performance on object recognition [28] and classification tasks in natural images [13]. In addition, CNNs have been used in biological applications such as restoring and segmenting volumetric electron microscopy images [16], [18], [19]. In this study, we propose to use CNNs on 3D optical microscopy images for neuron reconstruction. Through the proposed CNN, we obtained the prediction of neuron segmentation efficiently and accurately, which significantly improved the performance of neuron morphology reconstruction.

2.3 Fast prediction of CNNs

One major challenge of using CNNs on neuronal image segmentation is that the volumetric images usually have large sizes and thus could be computational expensive to segment. A common way of generating image segmentation using CNNs is to extract patches from images and use those patches as inputs to the trained network. The output of each patch is a single label of the center pixel of that patch. Such patch-based prediction results in a huge amount of redundant computations. It is thus desirable to design a fast prediction algorithm that can segment the whole image directly without generating patches.

In [29], a fast prediction algorithm was proposed by creating a group of fragments over the feature maps generated from each max-pooling layer. Each fragment contains information independent of other fragments, and the output fragments at the last layer were reorganized to generate the final prediction of the whole image. Although this fast prediction algorithm does not require patch extraction, some duplicate computations still exist when generating different fragments. In [30] the d-regularly sparse kernels were introduced to eliminate the redundant computations in convolutional and pooling layers. Those sparse kernels converted convolution and pooling kernels with various strides into operations with a single stride. This conversion ensured continuous memory access and increased the computational efficiency on GPUs. However, this sparse kernel prediction method is not applicable to architectures in which feature

maps are padded before each convolutional layer to retain their sizes.

In this study, we employed a novel network to obtain image predictions efficiently. In particular, we applied deconvolution operations to offset the size reduction caused by convolution and max-pooling operations. Contrary to the convolution operation that connects multiple input activations using a convolution kernel to produce a single output, the deconvolution associates a single input with multiple outputs using a deconvolution transformation. One significant advantage of using deconvolutional layers is that they allow the output feature map to have the same size as the input image by using carefully selected deconvolution kernels and stride sizes. Specifically, we used multiple deconvolution operations at different intermediate layers to ensure a same size for feature maps at those layers. The deconvolved same-size feature maps were then combined to form a multi-scale representation of the model input. Through such design, our network is capable of generating an end-to-end mapping between inputs and outputs and leads to dense pixel-wise prediction over images without any computational redundancy.

2.4 3D fully convolutional neural networks

Another major challenge of using CNNs for neuronal image segmentation is the large diversity of the images in the BigNeuron database. First, the complicated structures of neurons require the network to be able to learn features from multiple scales. For example, the recognition of neuron skeleton needs large filters, since neurons can be very thin yet projecting to long distance in 3D space. On the other hand, the fine neuronic structures like branches or bifurcations needs small filters to capture the local information in a small neighborhood. Second, images in BigNeuron datasets are contributed by different research groups and acquired using different imaging techniques on various species such as fruit fly, fish, etc. This results in a large amount of differences of neuron morphologies among these images. Hence, the network is expected to extract the essential common characteristics of neurons over those various samples. Third, the qualities of images in BigNeuron are dramatically different, which introduced an extra difficulty for the model training. For example, the signal levels of key neurite in some images might be strong and thus easy to detect, but these signals might be weak in other images because of multiple factors such as inappropriate exposure, noises from non-neuronal cells, or limitations of experimental devices.

To overcome the above-mentioned difficulties, we proposed a novel 3D CNN for reconstructing neurons accurately. Prior studies have demonstrated the superior performance of CNNs on classification about natural images. However, in those applications, 2D convolutions are applied to mimic the visual information processing. When applied to the neuron reconstruction problems, convolutions are expected to be 3 dimensional since neurons are naturally in 3D space. Architectures with 3D convolutions have been successfully used in video analysis [12], [31], [32], [33], in which the video data was viewed as a 3D volume where time acts as the third dimension. In [34], 3D filters were used to build nonlinear mappings between different image modalities, but its architecture was too shallow and inefficient for our neuron tracing task. In order to tackle the large variance of neuronal structures in 3D space, we proposed a fully convolutional network (FCN) with 3D convolutions to localize the neurons in the BigNeuron database. FCNs are variants of CNNs by replacing convolutional layers with fully connected layers. Such model allows the network to operate on inputs of any size and produce outputs of corresponding spatial dimensions. Fully convolutional layers together with deconvolutional layers ensure the feasibility of end-to-end training and testing. Furthermore, in order to improve the performance of networks, the strategy of stacking more layers is widely used in literature. Such strategy has two major drawbacks. One is deeper network typically means a larger number of parameters, which increases the risk of overfitting. The other drawback is the dramatically increased computational complexity. In this study, we proposed to use two powerful deep learning techniques introduced in the following to overcome these limitations.

2.5 Inception learning

One typical way for overcoming the above mentioned two limitations of deeper networks is to introduce small kernels instead of large kernels. However, large kernels are still necessary for capturing information of large regions in the input. In order to deal with this dilemma, inception networks [35] used multiple kernel sizes and max pooling in parallel in each stage, and then aggregated their outputs for the next stage. In each inception module, prior to expensive convolutions with large filter sizes, convolutions with small kernels are inserted to reduce the dimensionality such as the number of feature maps. An advantage of such design is that it allows for increasing the number of units at each stage significantly without an uncontrolled blow-up in computational complexity at later stages. Meanwhile, the use of multiple filter sizes ensures that the hidden information can be processed at various scales simultaneously.

2.6 Residual learning

The residual learning technique [36] was initially proposed to solve the degradation problem in which the training accuracy decreases when using too many stacked layers. Formally, we use $H(x)$ to denote the desired nonlinear mapping between the input feature map x and the output feature map after applying stacked layers. In residual learning, the stacked convolutional layers were considered to only fit $H(x)$'s nonlinear residual $F(x) := H(x) - x$. In other word, the mapping $H(x)$ can be reformulated as $F(x) + x$ and realized by adding a shortcut identity mapping connection between the input feature map and the output map. Such design ensures that residual networks can achieve more accurate results using deeper models but the training will be still very fast. This is because previous layers are reused by subsequent layers through shortcut connections, which makes each layer learns less than a plain network but utilized more information.

In this study, we used residual learning for segmenting the neuron morphology from optical microscopy images. In addition to its excellent performance in dealing with the

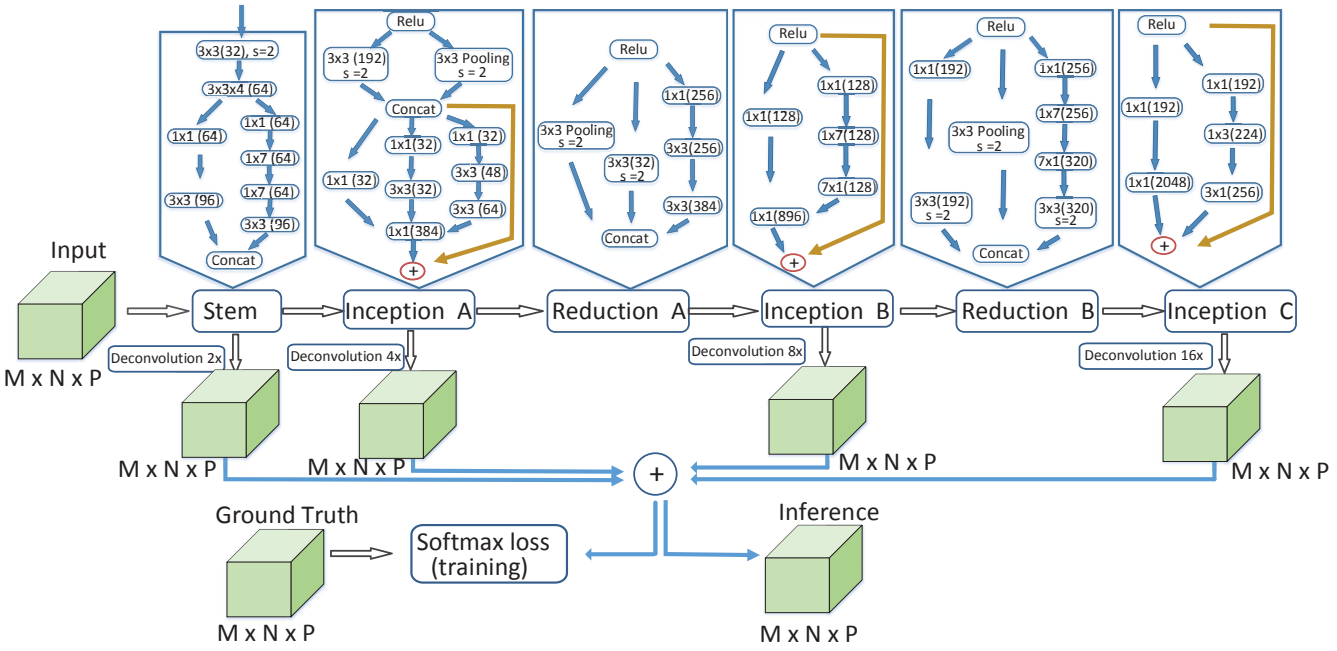

Fig. 1: Detailed architecture of the proposed method for dense prediction. For each module in the architecture, convolutional layers are denoted by filter sizes and number of feature maps in the brackets. Except for those layers with a stride size of 2, which are indicated by 's=2', the stride sizes in other layers are all 1. The filter sizes in the third dimension are all 1, and thus are skipped in the figure. The orange arrows indicate the shortcuts in residual learning.

degradation problem, another motivation of using residual learning is the limited number of samples for training. Although the number of images in the BigNeuron database is large, only a small number of them have manual reconstructions by experts. For segmentation tasks over 2D natural images, transfer learning could be used to borrow weights from pre-trained models on other similar data sets. However, transfer learning may not work in the segmentation of 3D neuron images because of the absence of pre-trained models for transferring. This lack of training samples restricted the depth of the model since deep models with limited samples may overfit. Due to the appealing property of residual learning, we employed shortcut connections between both convolutional and deconvolutional layers to construct the hierarchical representation of neurons.

2.7 The proposed deep model architecture

We provided the detailed configuration of our proposed deep network in Figure 1. The whole network contains 6 modules, and each of them used multiple paths for realizing inception learning. In particular, we used 3 modules (inception A, B, C in Figure 1) mainly for building the nonlinear relationship between input and output, and an additional 3 modules mainly for reducing the sizes of feature maps (reduction A, B, C in Figure 1). Multiple deconvolutional layers were used for up-sampling the intermediate feature maps to have the same size as the input. All outputs of deconvolutional layers were summed together to form a multi-scale representation of input. In those 3 inception modules, residual learning was applied for improving the performance and also reducing the computation complexity. The network is 48-layer deep when counting only layers with parameters. The overall number of layers (including scaling and batch normalization) used for the construction of the network is over 200. The whole network has 11,408,852 parameters in total.

All convolutional layers, including those inside inception modules, and deconvolutional layers used the rectified linear activation. Considering the limited number of foreground neuron voxels in original 3D images, we randomly sampled background voxels for each training subject in order to control the numbers of positive and negative samples. Intuitively, voxels around the neuron boundary are more difficult to classify correctly than other distant voxels. Therefore, most of background samples are drawn around the neuron boundary. Also the ratio between numbers of foreground and background voxels is kept within an acceptable level. Note that the numbers of voxels in foreground and background classes are not balanced. If the contribution of each class to the loss is treated equally during the training, the obtained network will mainly capture the discriminative features of the majority class which is the background class in our case. To this end, in the loss layer, we applied different weights to foreground and background classes to overcome this imbalance problem. To be specific, for each 3D training sample, we firstly computed individual percentages of foreground and background classes contained and then used the inverses of these percentages as weights of corresponding classes for computing the training loss. Through this way, the importance of majority class, i.e. the background class, is decreased and the training accuracy could be more meaningful for monitoring the network performance. We

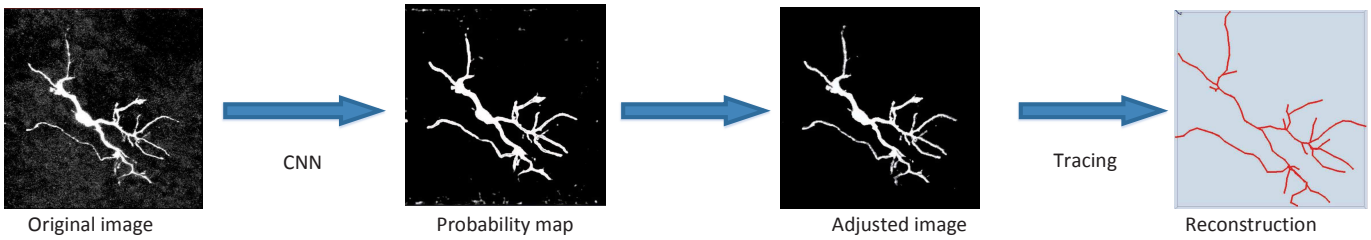

Fig. 2: The pipeline of our method for automated reconstruction of neuronal structures.

trained our deep network with stochastic gradient decent. We used a mini-batch size of 256, a momentum of 0.9 and a weight decay of 0.0005. The weights are initialized randomly following a gaussian distribution with zero mean and standard deviation of value 0.01. The biases were initialized to either 0 or 1. We firstly set the base learning rate to be 10^{-3} and then decreased its value following a polynomial decay for each iteration. We stopped the training when the relative accuracy difference between two consecutive iterations is less than 0.01. Experimentally, we found that the model requires around 37,000 iterations of training to achieve the desired accuracy using the publicly available package 'Caffe' [37] on an NVIDIA $K80$ GPU with 12GB memory. The whole training time on the data set containing augmented images took roughly 74 hours. For testing phase, the segmentation time varies from 20 seconds to 150 seconds on GPU for each subject, depending on the input image size. For example, when the input image size is $984 \times 980 \times 74$, the segmentation took 143 seconds using GPU.

2.8 Image adjustment

For each microscopy neuron image, we can generate its probability map P having the same size as the image. Each element of P indicates the probability of the corresponding voxel as neuron, known as foreground probability. A natural way to use the probability map is to apply the tracing algorithm directly to detect the neuronal structure. In this study, we combined the original image and the probability map together to get a combined representation for suppressing the noise signal effectively. Specifically, for an input image $I(x)$ where x is a voxel, we screened the probability map by a threshold δ chosen empirically to identify the foreground voxels. If the foreground probability $P(x)$ of the voxel x is less than δ, we set the intensity $I(x)$ to be zero, otherwise we keep its original intensity value unchanged. Thus, an intermediate intensity image $\tilde{I}(x)$ is defined as

$$\tilde{I}(x) = \begin{cases} I(x) & \text{if} \quad P(x) > \delta, \\ 0 & \text{if} \quad P(x) \leq \delta. \end{cases}$$

In order to further use the probability map, we constructed a new probability image $\tilde{P}(x)$ by multiplying the probability value $P(x)$ with 255 and then rounding the decimal value to the closest integer. The final adjusted image $F(x)$ by our method is written as

$$F(x) = \alpha \tilde{I}(x) + (1 - \alpha)\tilde{P}(x) \tag{1}$$

where α is a weight to control the contributions of the original intensity and the probability map. Then the tracing algorithm will be applied on the adjusted image to trace the final neuronal reconstruction. The detailed pipeline of the proposed method is illustrated in Figure 2.

3 Experimental Results

3.1 Experimental setup

In this study, we selected 68 subjects from BigNeuron database. These subjects are from a variety of species, and each subject has the corresponding expert manual reconstruction as our ground truth for training and evaluation. Out of these 68 3D images, more than half of them contain clearly visible noise in the images. We randomly sampled 3/4 of these 68 subjects for training the proposed models. Then the model was evaluated on remaining subjects to compute the neuron tracing performance. The training and testing split was performed in a stratified way in terms of data source to avoid the scenario that all subjects from some research labs are all either in training or test sets. The training patch size is selected as 160*160*8 considering computational resources and also unequal image sizes along three directions. For the testing phase, we used the whole testing image as the input, thus the input size would be changed according to the size of testing image.

During the training phase, we trimmed off the background margins of training images according to the ground truth for saving computational resources. In order to improve model performance, we used data augmentation to enlarge the training data set. The data augmentation includes transformations of original images with rotation and flipping along different dimensions. During the test phase, we empirically selected the probability threshold value δ in 1 to be 0.2. The coefficient α in Equation 1 was tuned through line-search over test images and the value of 0.85 was used. We applied three distance scores to measure the difference between a particular reconstruction and the ground truth. These scores were defined in [1] and are known as 'entire structure average', 'different structure average' and 'percentage of different structures'. Specifically, they are calculated in following ways. For each node in manual reconstruction, we calculated the minimal spatial distance between this node and all nodes in the reconstructed nodes generated by computational methods. The entire structure average is obtained by averaging all these reciprocal minimal spatial distances; the different structure average only sums those distances that are larger than 2 voxels since the difference of two nodes that is less than 2 voxels is hardly distinguished visually. The percentage of different structures captures the percentage of pairs of nodes whose

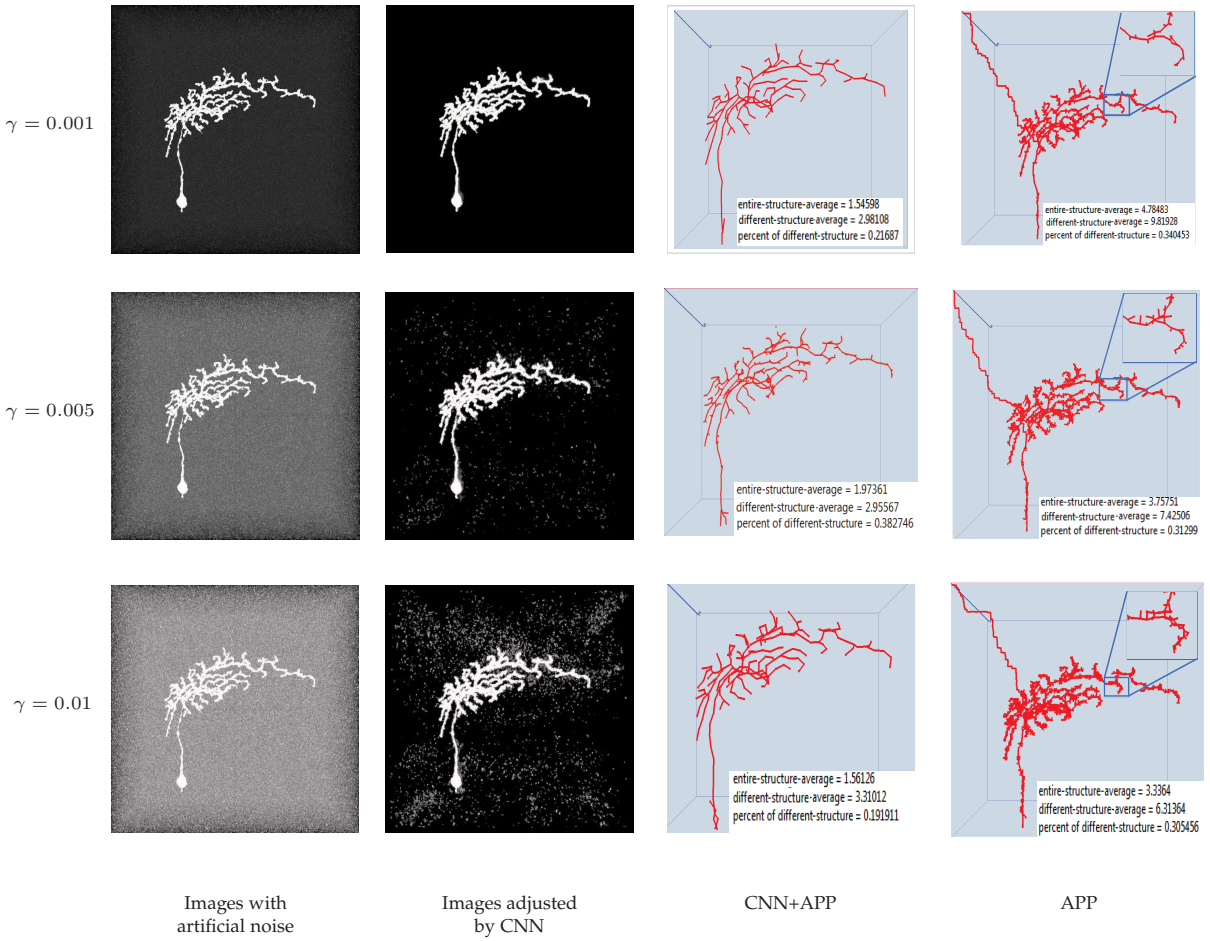

Fig. 3: Comparisons of tracing performance on images with artificial noises. The first column shows the raw images with different levels of added Gaussian white noises. The variance values are 0.001, 0.005, 0.01, respectively, from top to bottom. The second column shows the corresponding adjusted images after applying our CNN model to remove the noise voxels. The third column shows the reconstruction results by the APP method on the adjusted images. The fourth column shows the reconstruction results by the APP method on the contaminated images. The up-right boxes in the fourth column show the parts of the corresponding images. The distances between the tracing result and the ground truth is also give in each image in the third and fourth columns.

reciprocal minimal spatial distances are more than 2 voxels. For all of these three scores, larger values indicate higher discrepancy between the tracing results and the manual reconstruction. Note that the main contribution of our work is to provide a framework for enhancing the input quality of tracing methods. In order to demonstrate the performance improvement of our proposed method, we adapted APP and its variant APP2, since these methods are widely used and have been shown excellent robustness over noise [9]. We also chose SmartTracing in our experiments since it is also a learning-based method newly developed and has excellent performance when the noise and broken structures exist. We didn't show the results of other tracing methods because of the page limit and also the great popularity of the above 3 methods. Both APP and APP2 methods reconstruct the 3D morphology of a neuron generally in two steps. They firstly produce an initial over-reconstruction using shortest paths between pixels and seed locations by a fast-marching algorithm. Then, they refine the initial reconstruction by pruning redundant segments. Compared with APP, APP2 has some improvements from the following aspects. Firstly, APP2 proposes a gray-weighted image distant transformation on the raw image, which is absent in APP. Secondly, APP2 creates a parental map to generate shortest paths instead of using a large graph which appears in APP. Thirdly, APP2 employs a hierarchical pruning with a new coverage ratio computation formula motivated by APP. In addition, considering the unbalanced image sizes in different directions, i.e. the height and width of images are usually much larger than the depth, we showed the visual experimental results with almost front view. We believe using this angle can bring better visualizations about comparison results between different methods.

3.2 Performance on images with artificial noise

A major advantage of our proposed CNN model is that we can obtain accurate segmentations of neurons. Such segmentations can enhance the signal of neurons from a

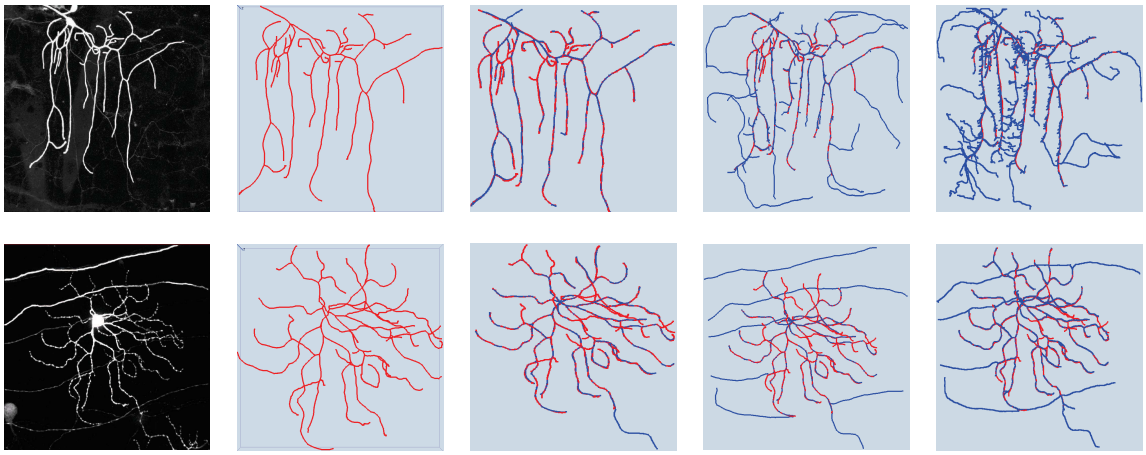

Fig. 4: Comparisons with different neuron tracing methods for two images in BigNeuron database. The first row shows results for the subject No. 31 and the second row is for the subject No. 52. The first column shows the original images. The second column shows the expert manual reconstruction results. The third column is the reconstructions using APP2 on adjusted images with our CNN model. The fourth column is the results by applying APP2 directly on the original images. The fifth column shows the results by SmartTracing method on the original images. Note that, for the last three columns, we overlayed computational reconstructions in blue with manual reconstruction results which are denoted by red.

noisy image, which improves the neuron tracing applied in the following. In order to demonstrate the robustness of our CNN models with respect to different levels of noises, we randomly select one testing raw image because of the page limit, and added Gaussian white noise of mean 0 and different variance γ. We then used each image contaminated with artificial noise as the input to our CNN model, and generated the corresponding adjusted image as described in Section 2.8. For both contaminated images and adjusted images by CNN models, we applied APP as the tracing method to reconstruct the neuron structures. In the following experiments, we mainly used APP2 as the tracing methods, since APP2 is more popular than APP even though they both used similar idea for tracing. The reason of not using APP2 here is that it couldn't generate visible results in a reasonable time when the level of noise is high. The visualization results for three values of $\gamma = 0.001$, 0.005 and 0.01 are given in Figure 3. Note that, the figures in this work are all shown in 3D space. This could be seen from the light blue axis shown in the above Figure 3.

We can observe that after applying our proposed segmentation model, the numbers of noise pixels in adjusted images are significantly decreased. The reconstructions on adjusted images are visually close to the ground truth by human experts. In contrast, the reconstructions on contaminated images without adjustments contain many unnecessary spurs because of the noise signals. In addition, we computed the distances between the reconstructions and the ground truth. We can see that for different levels of noise, the performance on adjusted images is consistently better than that on contaminated images. This shows that our trained CNN models could be potentially used to improve the accuracy of neuron tracing, especially if the microscopy images are contaminated by noise.

3.3 Performance on real images

In order to further demonstrate the advantage of our CNN models on denoising neuron microscopy images, we used two real images with visible noise contributed by different research labs. In addition, we compared our method with SmartTracing method [11] which is another learning-based tracing framework. We used APP2 as the tracing method applied on original and adjusted images, since SmartTracing is also based on APP2. The reconstruction results of different methods are given in Figure 4.

We can see that the tracing results on adjusted images are visually more similar to the ground truth than those on original images. The probability maps of our CNN model captured those bright noise voxels, which facilitated the reconstructions applied subsequently. In contrast, many redundant branches and bifurcations are still wrongly kept as neuron segments by APP2 on the original images. Although SmartTracing is also a learning based method, it used local features with complicated transformations to train the model. In contrast, CNNs employed stacked layers with different filter sizes to extract multi-scale information of objects in the image. The advantage of our model over SmartTracing can be observed in Figure 4 as well.

3.4 Comparison with other methods

In order to evaluate the proposed method quantitatively, we compared the tracing performance of APP2 and SmartTracing with and without adjustment by our CNN model respectively. The results on all 17 test subjects are reported in Table 1, 2, and 3 with different measures. We can observe that, for both APP2 and SmartTracing, their performance on adjusted images is better than that based on original images for all three measurements. Specifically, APP2 with CNN adjustment outperformed APP2 on original images for all subjects with respect to the measure 'entire structure

TABLE 1: Quantitative comparisons of tracing performance by APP2 and SmartTracing method with and without the adjustment by my CNN model respectively. The results reported in this table are based on the measure 'entire structure average'.

	entire structure average			
	CNN+APP2	APP2	CNN+ST	ST
Sub. 3	**1.4667**	19.1914	**1.9345**	13.253
Sub. 4	**1.4608**	20.4576	**2.2307**	N/A
Sub. 5	**2.9562**	3.4425	2.9451	**2.8185**
Sub. 7	**1.7715**	4.1832	**2.2249**	4.1815
Sub. 16	**1.7643**	2.7168	**2.1222**	2.3085
Sub. 17	**1.5133**	1.5218	**1.9180**	1.9883
Sub. 18	**1.9768**	2.1165	2.5506	**2.4962**
Sub. 19	1.7361	**1.6803**	2.0086	**1.9669**
Sub. 20	**2.2600**	4.4414	**2.5101**	3.0851
Sub. 21	**1.9406**	2.9079	**1.7357**	1.9998
Sub. 22	**2.0178**	2.1073	**2.0318**	2.0717
Sub. 25	**15.1363**	25.2285	**16.9098**	18.8010
Sub. 29	**2.3182**	4.3457	**2.5062**	4.2284
Sub. 31	**1.5740**	13.4657	**1.8931**	10.0305
Sub. 35	**3.0737**	31.0310	**3.4451**	N/A
Sub. 50	**8.6395**	21.3847	**7.8424**	13.3103
Sub. 52	**4.3104**	17.6192	**3.3240**	12.0861

TABLE 2: Quantitative comparisons of tracing performance by APP2 and SmartTracing method with and without the adjustment by my CNN model respectively. The results reported in this table are based on the measure 'different structure average'.

	different structure average			
	CNN+APP2	APP2	CNN+ST	ST
Sub. 3	**3.7271**	24.3327	**3.8273**	17.6276
Sub. 4	**2.9812**	24.4982	**3.6339**	N/A
Sub. 5	**5.7559**	6.6293	4.1684	**4.0008**
Sub. 7	**3.1334**	7.6208	**4.8516**	7.8795
Sub. 16	**4.2014**	6.9710	**4.2785**	5.2957
Sub. 17	**2.7030**	2.8417	**2.8988**	3.0038
Sub. 18	**3.1877**	3.3327	4.8305	**4.7152**
Sub. 19	**2.7909**	2.7932	3.0154	**2.9459**
Sub. 20	**3.2338**	6.3721	**3.9653**	5.1144
Sub. 21	**3.0792**	5.3018	**2.9771**	3.7801
Sub. 22	**3.3857**	3.5341	**3.2598**	3.5386
Sub. 25	**18.9222**	29.3581	**20.0443**	22.1727
Sub. 29	**3.7772**	7.1298	**4.4599**	8.1930
Sub. 31	**5.3058**	23.2479	**4.0464**	13.5121
Sub. 35	**6.6046**	32.0973	**5.2681**	N/A
Sub. 50	**9.8630**	22.5079	**9.1288**	14.4537
Sub. 52	**17.4785**	37.7065	**9.8060**	25.2226

average'. Even for other two measures, there are only one or two subjects on which APP2 without CNN yielded slightly better performance. Similar performance gains by our CNN models are also observed on the results using SmartTracing. In addition, we can see that for those images with a large amount of noise such as Subject 3, 4, 31, and 52, the tracing results after using our CNN models are significantly better than those on original images. These results further demonstrated that the proposed 3D CNN method is very effective in improving neuronal reconstruction on noisy images.

4 Conclusion and Future Work

In this study, we aimed at improving neuronal reconstruction based on microscopy images. This was achieved by employing novel deep architectures to segment the neuronal voxels. The CNNs used the original 3D microscopy images as input and generated the segmentation maps as output. We compared the performance of our approach with that of commonly used tracing methods. Results showed that our proposed model significantly outperformed prior methods on other neuron tracing methods on microscopy images. Overall, our results demonstrated that our proposed method produced more accurate results on neuronal morphology reconstruction.

In this study, we considered the CNN model for neuron segmentation. There are also other deep architectures that achieved promising performance on image-related tasks. It would be interesting to apply other deep models for 3D image segmentation. For example, recent studies showed that recurrent neuron networks (RNNs) yielded very promising performance on visual recognition tasks. We will explore RNNs in the field of neuronal reconstruction in the future. In the current experiments, we only used a small number of images for training the CNN models. Prior studies have shown that the success of deep learning models relies on a large training data set. As more and more data with manual reconstruction are collected in the BigNeuron project, we will explore CNNs with deeper architectures in the future. The current work used CNN models for neuronal image segmentation and improved the quality of subsequent reconstruction. In the future, we will explore the possibility of using CNNs on neuron tracing directly.

TABLE 3: Quantitative comparisons of tracing performance by APP2 and SmartTracing method with and without the adjustment by my CNN model respectively. The results reported in this table are based on the measure '% of different structure'.

	% of different structure			
	CNN+APP2	APP2	CNN+ST	ST
Sub. 3	**0.2075**	0.5347	**0.3153**	0.5339
Sub.4	**0.2226**	0.7012	**0.4178**	N/A
Sub.5	**0.3472**	0.3483	0.5080	**0.4974**
Sub.7	**0.2178**	0.3920	**0.2686**	0.3629
Sub.16	**0.2099**	0.2916	0.2488	**0.2204**
Sub.17	0.2096	**0.2019**	**0.2968**	0.3134
Sub.18	**0.3353**	0.3645	0.3125	**0.3023**
Sub.19	0.3010	**0.2748**	0.3276	**0.3036**
Sub.20	**0.4588**	0.5684	**0.4092**	0.4130
Sub.21	**0.3498**	0.4163	**0.2346**	0.2446
Sub.22	**0.3298**	0.3474	0.3188	**0.2901**
Sub.25	**0.5443**	0.5727	0.6387	**0.6101**
Sub.29	**0.3663**	0.5017	**0.3452**	0.3948
Sub.31	**0.1683**	0.3807	**0.2483**	0.4712
Sub.35	**0.3622**	0.7447	**0.5120**	N/A
Sub.50	**0.8442**	0.9089	**0.8286**	0.9116
Sub.52	**0.2492**	0.4283	**0.2723**	0.3845

Acknowledgments

This work was supported in part by National Science Foundation grant DBI-1641223, Old Dominion University, and Washington State University. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research.

References

[1] H. Peng, Z. Ruan, D. Atasoy, and S. Sternson, "Automatic reconstruction of 3D neuron structures using a graph-augmented deformable model," *Bioinformatics*, vol. 26, no. 12, pp. 38–46, 2010.

[2] Y. Wan, F. Long, L. Qu, H. Xiao, M. Hawrylycz, E. W. Myers, and H. Peng, "BlastNeuron for automated comparison, retrieval and clustering of 3d neuron morphologies," *Neuroinformatics*, vol. 13, no. 4, pp. 487–499, 2015.

[3] U. Sümbül, A. Zlateski, A. Vishwanathan, R. H. Masland, and H. S. Seung, "Automated computation of arbor densities: a step toward identifying neuronal cell types," *Frontiers in Neuroanatomy*, vol. 8, p. 139, 2014.

[4] U. Sümbül, S. Song, K. McCulloch, M. Becker, B. Lin, J. R. Sanes, R. H. Masland, and H. S. Seung, "A genetic and computational approach to structurally classify neuronal types," *Nature Communications*, vol. 5, 2014.

[5] L. Gu and L. Cheng, "Learning to boost filamentary structure segmentation," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 639–647.

[6] T. Smafield, V. Pasupuleti, K. Sharma, R. L. Huganir, B. Ye, and J. Zhou, "Automatic dendritic length quantification for high throughput screening of mature neurons," *Neuroinformatics*, vol. 13, no. 4, pp. 443–458, 2015.

[7] Z. Zhou, S. A. Sorensen, and H. Peng, "Neuron crawler: an automatic tracing algorithm for very large neuron images," in *Proceeding of International Symposium on Biomedical Imaging (ISBI)*, 2015, pp. 870–874.

[8] Z. Zhou, X. Liu, B. Long, and H. Peng, "TReMAP: Automatic 3d neuron reconstruction based on tracing, reverse mapping and assembling of 2d projections," *Neuroinformatics*, vol. 14, no. 1, pp. 41–50, 2016.

[9] H. Xiao and H. Peng, "APP2: automatic tracing of 3d neuron morphology based on hierarchical pruning of a gray-weighted image distance-tree," *Bioinformatics*, vol. 29, no. 11, pp. 1448–1454, 2013.

[10] J. Yang, P. T. Gonzalez-Bellido, and H. Peng, "A distance-field based automatic neuron tracing method," *BMC bioinformatics*, vol. 14, no. 1, pp. 1–11, 2013.

[11] H. Chen, H. Xiao, T. Liu, and H. Peng, "SmartTracing: self-learning-based neuron reconstruction," *Brain Informatics*, vol. 2, no. 3, pp. 135–144, 2015.

[12] S. Ji, W. Xu, M. Yang, and K. Yu, "3D convolutional neural networks for human action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 221–231, 2013.

[13] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., 2012, pp. 1106–1114.

[14] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proceedings of the European Conference on Computer Vision*. Springer, 2014, pp. 818–833.

[15] T. Zeng, R. Li, R. Mukkamala, J. Ye, and S. Ji, "Deep convolutional neural networks for annotating gene expression patterns in the mouse brain," *BMC bioinformatics*, vol. 16, no. 1, pp. 1–10, 2015.

[16] D. C. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Mitosis detection in breast cancer histology images with deep neural networks," in *Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention*, vol. 2, 2013, pp. 411–418.

[17] Y. LeCun, F. J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2004, pp. II–97.

[18] V. Jain and S. Seung, "Natural image denoising with convolutional networks," in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., 2009, pp. 769–776.

[19] S. C. Turaga, J. F. Murray, V. Jain, F. Roth, M. Helmstaedter, K. Briggman, W. Denk, and H. S. Seung, "Convolutional networks can learn to generate affinity graphs for image segmentation," *Neural Computation*, vol. 22, no. 2, pp. 511–538, 2010.

[20] W. Zhang, R. Li, H. Deng, L. Wang, W. Lin, S. Ji, and D. Shen, "Deep convolutional neural networks for multi-modality isointense infant brain image segmentation," *NeuroImage*, vol. 108, pp. 214–224, 2015.

[21] A. Fakhry, H. Peng, and S. Ji, "Deep models for brain EM image segmentation: novel insights and improved performance," *Bioinformatics*, vol. 32, pp. 2352–2358, 2016.

[22] R. Li, D. Si, T. Zeng, S. Ji, and J. He, "Deep convolutional neural networks for detecting secondary structures in protein density maps from cryo-electron microscopy," in *Proceedings of the IEEE International Conference on Bioinformatics and Biomedicine*, 2016, pp. 41–46.

[23] A. Fakhry, T. Zeng, and S. Ji, "Residual deconvolutional networks for brain electron microscopy image segmentation," *IEEE Transactions on Medical Imaging*, vol. 36, no. 2, pp. 447–456, 2017.

[24] H. Peng, M. Hawrylycz, J. Roskams, S. Hill, N. Spruston, E. Meijering, and G. A. Ascoli, "BigNeuron: large-scale 3d neuron reconstruction from optical microscopy images," *Neuron*, vol. 87, no. 2, pp. 252–256, 2015.

[25] Y. Liu, "The diadem and beyond," *Neuroinformatics*, vol. 9, no. 2, pp. 99–102, 2011.

[26] H. Peng, E. Meijering, and G. A. Ascoli, "From diadem to bigneuron," *Neuroinformatics*, vol. 13, no. 3, pp. 259–260, 2015.

[27] H. Peng, Z. Ruan, F. Long, J. H. Simpson, and E. W. Myers, "V3D enables real-time 3d visualization and quantitative analysis of large-scale biological image data sets," *Nature Biotechnology*, vol. 28, no. 4, pp. 348–353, 2010.

[28] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, November 1998.

[29] A. Giusti, D. C. Ciresan, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Fast image scanning with deep max-pooling convolutional neural networks," in *Proceedings of the 20th IEEE International Conference on Image Processing*, 2013, pp. 4034–4038.

[30] H. Li, R. Zhao, and X. Wang, "Highly efficient forward and backward propagation of convolutional neural networks for pixelwise classification," *arXiv preprint arXiv:1412.4526*, 2014.

[31] V. Badrinarayanan, A. Handa, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling," 2015. [Online]. Available: http://arxiv.org/abs/1505.07293

[32] H. Chen, X. Qi, J.-Z. Cheng, and P.-A. Heng, "Deep contextual networks for neuronal structure segmentation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016.

[33] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 1725–1732.

[34] R. Li, W. Zhang, H.-I. Suk, L. Wang, J. Li, D. Shen, and S. Ji, "Deep learning based imaging data completion for improved brain disease diagnosis," in *Proceedings of Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2014, pp. 305–312.

[35] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[37] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.