Bioinformatics, 32(15), 2016, 2352–2358
doi: 10.1093/bioinformatics/btw165
Advance Access Publication Date: 25 March 2016
Original Paper

Bioimage informatics

Deep models for brain EM image segmentation: novel insights and improved performance

Ahmed Fakhry[1], **Hanchuan Peng**[2] **and Shuiwang Ji**[3,*]

[1]Department of Computer Science, Old Dominion University, Norfolk, VA 23529, USA, [2]Allen Institute for Brain Science, Seattle, WA 98103, USA and [3]School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164, USA

*To whom correspondence should be addressed.

Associate Editor: Robert Murphy

Received on December 10, 2015; revised on March 3, 2016; accepted on March 5, 2016

Abstract

Motivation: Accurate segmentation of brain electron microscopy (EM) images is a critical step in dense circuit reconstruction. Although deep neural networks (DNNs) have been widely used in a number of applications in computer vision, most of these models that proved to be effective on image classification tasks cannot be applied directly to EM image segmentation, due to the different objectives of these tasks. As a result, it is desirable to develop an optimized architecture that uses the full power of DNNs and tailored specifically for EM image segmentation.
Results: In this work, we proposed a novel design of DNNs for this task. We trained a pixel classifier that operates on raw pixel intensities with no preprocessing to generate probability values for each pixel being a membrane or not. Although the use of neural networks in image segmentation is not completely new, we developed novel insights and model architectures that allow us to achieve superior performance on EM image segmentation tasks. Our submission based on these insights to the 2D EM Image Segmentation Challenge achieved the best performance consistently across all the three evaluation metrics. This challenge is still ongoing and the results in this paper are as of June 5, 2015.
Availability and Implementation: https://github.com/ahmed-fakhry/dive
Contact: sji@eecs.wsu.edu

1 Introduction

Anatomical connections between neurons in the brain form circuits that are responsible for the rapid information flow. Knowledge of the circuit structure is crucial for the investigation of its function. Mapping the structure and components of these circuits is one of the top priority research areas in neuroscience. It provides a foundation for understanding what the brain is made of at the cellular and structural levels, and how these properties change across the normal life span and in brain disorders. The reconstruction of such circuits at a very high resolution using electron microscopy (EM) is considered to be the gold standard for circuit mapping (Brain Research through Advancing Innovative Neurotechnologies, BRAIN, 2014).

Currently, sparse circuit reconstruction has been widely used on a small-scale. Most of the studies focused on the very small *Caenorhabditis elegans* or small parts of the nervous systems of the *Drosophila*. Recently, some of those efforts were extended to reconstruct the inner plexiform layer in the mouse retina (Briggman *et al.*, 2011; Helmstaedter *et al.*, 2013; Kim *et al.*, 2014). Using EM data in large-scale studies are currently a challenge, where the main bottleneck is data analysis. Better automatic image segmentation techniques would substantially amplify the impact of dense EM reconstruction. Machine learning and artificial intelligence approaches are expected to be the main driver for the desired advancements in this area.

In this work, we focused on the automatic segmentation of serial-section Transmitted Electron Microscopy (ssTEM) images. We provided a novel design of deep neural networks (DNNs) (LeCun *et al.*, 1989) that extends the techniques described in Ciresan *et al.*

(2012a). We built a pixel DNN classifier that predicts the probability of every individual pixel in a given image being a membrane (border) pixel or not. Our DNN classifier accepts raw pixel intensities as input without any preprocessing and learns highly discriminative features automatically before producing final probability maps. These probability maps were fed later to another machine learning classifier based on random forests (Breiman, 2001) to produce final segmentations.

DNNs have been widely used in a number of applications in computer vision. It achieved the state-of-the-art performance on tasks like large-scale image and video recognition (Ji *et al.*, 2013; Krizhevsky *et al.*, 2012; Zeiler and Fergus, 2014), digit recognition (Ciresan *et al.*, 2012b) and object recognition tasks (LeCun *et al.*, 2004). Recently, many attempts have been made to extend the usage of these models to the field of image segmentation, leading to improved performance (Briggman *et al.*, 2009; Jain and Seung, 2009; Jain *et al.*, 2007; Ronneberger *et al.*, 2015; Turaga *et al.*, 2010; Zhang *et al.*, 2015). Although some of the existing popular models have proved the ability to generalize well for different recognition tasks like the model in Krizhevsky *et al.* (2012), most of these models that excelled on image classification and recognition tasks cannot be applied directly to EM image segmentation tasks, given the difference in objectives between those tasks as well as the difference between EM and natural images (more details in Section 2.1). As a result, it is desirable to develop an optimized architecture that utilizes the full power of DNNs and tailored specifically for EM image segmentation.

In this work, we developed a DNN model architecture that is highly optimized for ssTEM image segmentation. The key contribution is in the model itself and the novel insights about the specific kernel configuration leading to substantially improved results. We evaluated the effect of model configuration along with kernel structures and depth on the final segmentation outcome. We validated our approach by applying it to the ISBI 2012 EM Segmentation Challenge (ISBI, 2012) (http://brainiac2.mit.edu/isbi_challenge/), achieving the best performance on all evaluation metrics out of more than 40 participating groups (more groups are participating as the challenge is ongoing). Our model was one of the few that were able to beat the performance of a second human observer.

2 Methods

In our work, we used a full stack of EM image slices of the *Drosophila* first-instar larva ventral nerve cord (Cardona *et al.*, 2010) provided by the organizers of the ISBI 2012 EM Segmentation Challenge (ISBI, 2012). The training stack consists of 30 grayscale sections of 512 × 512 pixels each, where there is a corresponding label map for each image slice representing whether the pixels are membrane or non-membrane. We trained a deep convolutional neural network (DNN) pixel classifier to predict the label of every pixel separately. We applied this classifier on another stack of 30 sections representing the testing data where the ground truth is only known to the organizers of the challenge.

In order to use a pixel classifier, we adopted a patch-based training technique. For every pixel in every slice, we extracted a square patch of a fixed size with the target pixel in its center. For boundary pixels, we mirrored the pixels across the slice borders. Upon testing, we obtained a membrane probability map for each slice of the testing data. These probability maps then underwent post-processing to generate the labels.

2.1 Key insights

The key contributions of this work are the optimized architecture of a DNN model for EM image segmentation and the underlying motivation and observations. Although the use of DNN architecture usually leads to good performance on similar segmentation tasks, a careful design of the network architecture and choices of kernel sizes and placement are the key to utilize the full performance power of the model. For example, in the model suggested by Krizhevsky *et al.* (2012), the kernel sizes were chosen to be very large at the bottom layers of the network and then reduced gradually. Later, Zeiler and Fergus (2014) showed that a better performance could be obtained by reducing the receptive field size and choosing a smaller stride for the first convolutional layer in the same model. On the other hand, in Simonyan and Zisserman (2014), a network of a very small kernel size which was fixed for all layers proved to achieve the best performance on image localization and classification tasks. This highlights the importance of these architectural details in achieving record-breaking performance on different computer vision tasks.

Unfortunately, the application of the very powerful models like Simonyan and Zisserman (2014) and Szegedy *et al.* (2014) to the task of EM image segmentation does not yield as good results as it did on natural image recognition tasks. The reason is that the two tasks are genuinely different in terms of their objective and training characteristics. In segmentation, the objective is to assign a label for every single pixel in the image as opposed to a single label assigned to the entire image in classification. In addition, training for segmentation tasks includes a lot of redundancy in the input data due to the patch-based technique, as opposed to training on the whole image or random crops of it in classification tasks. In terms of the data itself, EM images are characterized by their high density and the invariability of the objects it composes unlike the natural images that are regularly used in classification tasks like the ImageNet data (Deng *et al.*, 2009). We experimented with several networks that are considered state-of-the-art for image classification and recognition tasks and the results were inferior. In particular, the VGG net (Simonyan and Zisserman, 2014) has performed very poorly on this specific segmentation task. We attribute this to the very small kernel sizes (3 × 3) in the network which does not include enough contextual information that this segmentation task requires based on our experiments.

The key observation about this task is how important the context information is for building discriminative features especially in the bottom layers of DNN in EM image segmentation tasks. It is crucial to provide each kernel with a large enough receptive field especially in the bottom layers in order to be able to learn better features. At the same time, if the context is too wide for the bottom layers, the performance will drop due to the excessively large neighborhood which will contain some noise. For example, in the VGG-net (Simonyan and Zisserman, 2014) all the kernels are fixed to a size of 3 × 3 which is very small and not enough for the features to be learned from the underlying data. In Krizhevsky's network, they started with an excessively large kernel size of 11 × 11 at the bottom. When we tried both networks on the EM data, the results were both inferior. In our architecture, we focused on starting with a kernel size of 8 × 8 at the bottom which is moderately large and sufficient for the network to be able to learn discriminative features without being affected by noise. Increasing the kernel sizes of the bottom layers in the network increases the receptive fields of each unit in the resulting feature maps, thereby increasing the impact of context information in generating these features. This leads to learning more discriminative features which improves the overall performance. On the contrary, the pixels in the feature maps of the top layers already correspond to a very large receptive field due to the presence of max pooling layers beneath them regardless of the kernel sizes in the layer directly beneath them as illustrated in Figure 1. Using smaller kernel sizes for the upper layers also allows the model to grow deeper as we can add more convolution layers. As a result, we decreased the kernel sizes gradually as we went deeper in the network.

Another key insight is the impact of non-linearity and network depth on the overall network performance. We argue that increasing the number of convolution layers along with their corresponding rectified linear unit (RELU) layers usually increases the networkes accuracy. Increasing the number of convolution layers increases the number of features to be learned, while the RELU layers are responsible for increasing the non-linearity in the network and preventing the gradient from saturation. In Krizhevsky network (Krizhevsky *et al.*, 2012), although the input image size was 224 × 224, the network did not have enough non-linearity with only five convolution layers and five corresponding RELU layers in between them. On the other hand, our best performing network had six convolution layers while the input size is only 95 × 95. This high non-linearity in our network was crucial for obtaining a better performance. We validated these insights through the network design in the next section.

2.2 The architecture

We used DNN with multiple convolution, pooling and fully connected layers for our pixel classifier. We experimented with a wide range of window sizes, network depth and kernel sizes to assess the effect of each parameter on the final segmentation outcome. We used window sizes ranging between 35 and 95, depth between 6 and 8 trainable layers and kernels between 3 × 3 up to 10 × 10.

Although our architectures are quite different from each others, they share some common properties that experimentally proved to be the best for the current task. No preprocessing was applied to any of the networks except for mean subtraction. The overall mean of all the pixels in all sections was subtracted from each pixel value. In our designs, we limited the number of max pooling layers in favor of more convolution layers in all the architectures. Increasing the number of convolution layers helped the model find more discriminative features. The key challenge was to add as many convolution layers as possible without losing the translation invariance advantage that the max pooling layers provide. In addition, the choice of the window size would always affect the network depth and the number of convolution layers in turn. We compared several architectures containing different numbers of max pooling layers ranging between two and five and we found that three max pooling layers always give the best results. As a result, each network we trained is divided into three blocks; each block contains a number of convolution layers (differs per network configuration and per block) which are followed by a single max pooling layer of size 2 × 2 and stride 2 by the end of the block.

We also introduced back to back convolution layers interleaved by only RELU as the non-linear transformation in all architectures. Instead of using a single convolution layer with a very large kernel size in every block, we chose to stack multiple convolution layers above each other with moderate kernel sizes while adding RELU layers in between them. This was done mainly to increase the nonlinearity in the model and thus encouraging it to learn more complex features. In addition, breaking down a single large kernel into several smaller ones reduced the total number of parameters need to be trained, thus reducing the overall computation time (Simonyan and Zisserman, 2014).

Our networks concluded with two fully connected layers after their third block. The last fully connected layer contains only two neurons corresponding to the segmentation tasks. The outcomes of these two neurons were finally passed through a softmax layer to produce probability values that represent either membrane or non-membrane classes.

We trained four DNNs sharing the common characteristics described earlier but with different configurations. These architectures were inspired by the networks used in Ciresan *et al.* (2012a); Krizhevsky *et al.* (2012); Simonyan and Zisserman (2014). The full architectures of the four networks can be found in Table 1.

Networks A and B are both shallow with six trainable weight layers each. Network A has a very small window size of 35 × 35 pixels which highly restricted the depth and the kernel sizes used. Network B starts with a window size of 65 × 65 pixels which we utilized to test the effect of using excessively large kernels while keeping the depth constant. On the other hand, Networks C and D have a relatively large window size of 95 × 95 pixels each to include more contextual information and to allow them to grow deep. Both networks use slightly larger kernels than the ones used in network A and slightly less than the ones used in network B.

In our experiments, we hypothesized that starting with a large kernel size in the bottom layers of the network and reducing the size as we move upwards is much better than the opposite direction where we start with small kernels at the bottom. We tested this hypothesis through the configuration of networks C and D. Network C started with a small kernel size for its bottom layer, and then the

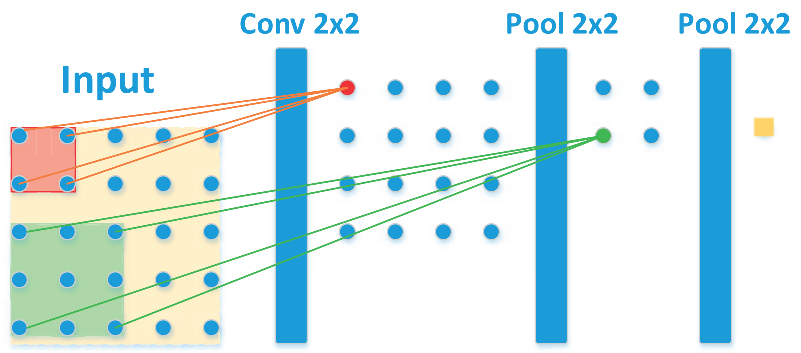

Fig. 1. In this figure, we demonstrate that the upper layers in the network always correspond to a large receptive field due to the presence of max pooling layers. In the figure, even with a shallow network that uses 2 × 2 convolution for its bottom layer and only 2 max pooling layers, the generated feature map in the last layer corresponds to a receptive field equals to the entire input size

Table 1. The complete architecture of the four DNNs used in our experiments

A	B	C	D
input 35 × 35	input 65 × 65	input 95 × 95	input 95 × 95
conv4-30	conv10-30	conv4-48	conv8-48
		LRN	LRN
maxpool			
conv3-50-pad1	conv5-50	conv5-128	conv6-128
		conv5-128	conv6-128
		LRN	LRN
maxpool			
conv3-60-pad1	conv3-60	conv6-256	conv4-256
conv3-60-pad1	conv3-60	conv6-256	conv3-256
		conv6-256	conv3-256
maxpool			
FC-100	FC-100	FC-500	FC-500
		Drop-0.5	Drop-0.5
FC-2			
softmax			

Each of the networks from A to D is divided into three blocks ending with a max pooling layer each. The convolution layer parameters are illustrated as 'conv <kernel size>-<number of kernels>-<padding size if any>'. All the max pooling layers are 2 × 2 with stride 2 while all the convolution layers are of stride 1. The RELU layers are omitted here for brevity.

kernel size was constantly increased till the top convolution layer. For network D, we started by a moderately large kernel size which is double the initial kernel size of network C then we constantly decreased it. Another advantage of beginning with a large kernel size was that we aggressively reduced the resolution of the feature maps, leading to a reduction in the computations.

To ensure learning in networks C and D, we applied local response normalization (LRN) before the max pooling layers in the first two blocks. This was because the number of parameters to be trained for these networks is higher than the other two due to the larger kernel sizes used and the increasing number of feature maps. Although RELUs are not easily saturated, adding LRNs is supposed to increase model generalization (Krizhevsky *et al.*, 2012). Experimentally, we found that the performance improvement using LRNs is minimal and does not contribute to the overall performance gain. We still included it in our models for the sake of completeness and to allow reproducibility of the results. We also applied dropout after the first fully connected layer in both networks C and D (details given in Section 2.4).

2.3 Test data augmentation

Ensemble learning techniques are well known to improve performance on various learning tasks. Random forest (Breiman, 2001) is a clear example that proved to be one of the most effective ensemble learning techniques. In random forests, many decision trees are built based on some random variations in the input and feature spaces, and eventually each decision tree votes for a specific class. The combined vote is then considered as the output decision of the random forest. Extending a similar scheme to neural networks is however computationally expensive. A single DNN usually takes several days of training even using GPU-based implementation for a data set with a million of samples. Training several of these networks would be computationally inefficient.

We chose to perform augmentation upon testing instead of the computationally expensive ensemble of DNNs [similar to the technique used in Cireşan *et al.* (2013)]. We applied several linear transformations on the input image before testing. The transformations were combinations of horizontal and vertical mirroring, and/or rotations by +90, −90 and 180°. After passing the transformed image through the network and obtaining a probability map, a reverse transformation was applied. In total, eight variations were applied to each testing image and then we took the average. The augmentation was implemented so that each variation received a vote in the final decision produced by the model. This technique is computationally more efficient than training several models as testing time is typically very fast. Our experiments showed a considerable advantage of applying those variations as the segmentation error dropped by half as compared with its original value.

2.4 Overfitting reduction

One of the challenges of using a DNN is overfitting. Because our deep networks have a huge number of parameters (up to tens of millions of them), we needed a very large data set to avoid overfitting. In our data set, we extracted all the patches of the membrane pixels in every slice and an equivalent number of non-membrane pixels sampled randomly per slice. This generated about 3 million training patches in total, a number that may not be enough to train a very deep network without the risk of overfitting.

We applied data augmentation to increase the variability in our training data. At the beginning of each epoch, a linear transformation to the input patch was randomly selected. We rotated each patch either by +90 or −90 and/or mirrored it either horizontally or vertically. The data augmentation significantly improved the accuracy of the classifier. In addition, we applied dropout with 0.5 dropping ratio after the first fully connected layer in networks C and D to further improve the performance.

2.5 Model design and implementation

Our patch-based classifier implementation is based on the publicly available C++ Caffe (Jia *et al.*, 2014) (branched out in October 2014) with several modifications. We implemented our custom data augmentation where we decoupled cropping from mirroring as it is not desirable in our experiments in addition to implementing rotations with several angles. We also added random shuffling of data at the beginning of each epoch through randomly dropping a few samples in each mini-batch with a specific percentage. A window size can also be supplied before augmentation to determine the patch size desired for each configuration to avoid the recalculation of the patches and the databases.

We trained our DNN classifier using back propagation (LeCun *et al.*, 2012) with stochastic gradient decent. We used a mini-batch size of 256, a momentum of 0.9 and a weight decay of 0.0005. We started with a base learning rate of 10^{-2} and decreased it by a factor of 10 every 100K iterations. We used random initialization for the weights through sampling from a normal distribution with a zero mean and 10^{-2} variance. The biases were initialized to either 0 or 1. Experimentally, we found that the model requires 30 epochs of training to achieve the desired accuracy on an NVIDIA K80 GPU. The training time took typically several days to complete.

We used an image-based approach for generating the probability maps for the testing data set. We implemented our own image-based forward propagation code (Giusti *et al.*, 2013). The model was trained first by Caffe using a patch-based approach then the weights of the kernels and biases were extracted and fed to our image-based code. Our image-based prediction speeded up the testing time dramatically as compared to a patch-based GPU forward propagation even though our code was running on a CPU (takes roughly 2–4 min on a CPU machine). In addition, our image-based code did not require any computational and storage overhead for generating the patches for the testing data, making it much more convenient and efficient.

2.6 Post-processing

Our network D achieved the best pixel accuracy on the ISBI 2012 data set without any post-processing. However, the best pixel accuracy is not necessarily accompanied with the best segmentation. Even the slightest mis-prediction of certain boundary pixels could severely hurt the overall segmentation. Our model is in nature a pixel classifier that was designed to achieve the best pixel accuracy but was not optimized to produce a better segmentation.

We used the watershed merge tree post-processing technique used in Liu *et al.* (2012, 2013). The technique starts by generating an initial segmentation using watershed for each probability map and then gradually raising the water level to produce hierarchical segmentations forming a watershed merge tree. A decision of merging two nodes in the merge tree is based on the result of a random forest boundary classifier that predicts the merge based on various features extracted from each two nodes. This scheme reduced the segmentation error to less than half of its original value.

3 Results and discussion

We created a validation data set from the training slices for which the truth labels are available. We divided the 30 training slices into 20 training and 10 validation slices to obtain quantitative evaluations for our models before submitting the final results to the ISBI 2012 challenge. The evaluation of the testing data in this challenge was done through an automated online system where the submitted segmentations were compared with the hidden ground truth based on three different metrics:

Minimum Splits and Mergers Warping error is a segmentation metric that penalizes topological disagreements, i.e.: the number of splits and mergers required to obtain the desired segmentation.

Foreground-restric Rand error is defined as 1 − the maximal F-score of the foreground-restricted Rand index, a measure of similarity between two segmentations.

Pixel error is defined as 1 − the maximal F-score of pixel similarity, or squared Euclidean distance between the original and the result labels.

We note that these evaluation metrics have been modified by the challenge organizers after this manuscript has been submitted. We demonstrate most of the results and model comparisons on the validation data before we show the performance of the best model on the testing data. It is worth mentioning that the results based on the validation data set may be affected by the reduction in the training data set size. This is because we only trained on the 20 slices instead of all the 30 slices, since the remaining 10 slices are in the validation set. Nonetheless, the results obtained on the validation data set are very useful for model comparison. Table 2 illustrates the comparison between the four networks evaluated on the validation data set. Although all the pixel error values are similar, the rand error gives a better interpretation in terms of segmentation. We observed that the post-processing reduced the rand error, but increased the pixel and warping errors. This is because the DNN is a pixel classifier aiming at optimizing the pixel error directly, and the warping error is closely related to the pixel error. Thus, we chose to report the pixel and warping errors before we applied the post-processing. After post-processing, the pixel and warping errors were highly altered in favor of obtaining a better segmentation. For that, the rand error values reported in Table 2 are obtained after applying the post-processing.

Table 2. Comparison between the four networks using the validation data set

Network	Rand error [0.10^{-3}]	Warping error [0.10^{-6}]	Pixel error [0.10^{-3}]
A	82	1548	52
B	95	1729	50
C	75	1775	51
D	47	1684	49

Both the pixel and warping errors are reported before applying the post-processing while the rand error is reported after the post-processing.

Table 3. Comparison between the results obtained using network D before and after using train data augmentation and test data augmentation

Network	Rand error [0.10^{-3}]	Warping error [0.10^{-6}]	Pixel error [0.10^{-3}]
D	47	1684	49
D—no training augmentation	271	2905	61
D—No testing augmentation	212	2176	61

These results were obtained by evaluation on the validation data set.

Our results are clearly in favor of the deeper networks C and D, which highlights the impact of the depth on the outcome of the segmentation. Networks A and B used less number of trainable layers, and this hurt the performance regardless of the kernel sizes used or the model configuration. On the other hand, networks C and D used a large window size with a deeper architecture and relatively large kernel sizes. Network D in particular achieved the best performance among all the other networks in terms of rand index. This validates our hypothesis that starting with a large kernel size in the bottom layers indeed helps the model learn more useful features than starting with small kernels as in network C.

We report the effectiveness of the techniques described in Sections 2.3 and 2.4 in Table 3. The comparison was made using our best performing network D on the validation data set. We noticed that both the data augmentation and rotations during testing are highly effective for improving the segmentation accuracy. The model suffered from severe overfitting without augmentation while the rotations applied during testing provided robustness to the classifier, thereby reducing the effect of input data randomization to a great extent.

After selecting the best architecture and evaluating different techniques applied upon training and testing, we extended our experiments to the testing data. We trained network D on the 30 training slices and evaluated them using the online system for the ISBI 2012 Challenge. Note that, although winners have been declared before ISBI 2012, the challenge is still ongoing. Figure 2 reports the final results on two slices of the testing stack.

On the ISBI 2012 Challenge data set, our approach achieved the best results on all the three evaluation metrics, breaking the record maintained by other participants for years. A complete comparison between our approach and other competing methods is shown in Table 4.

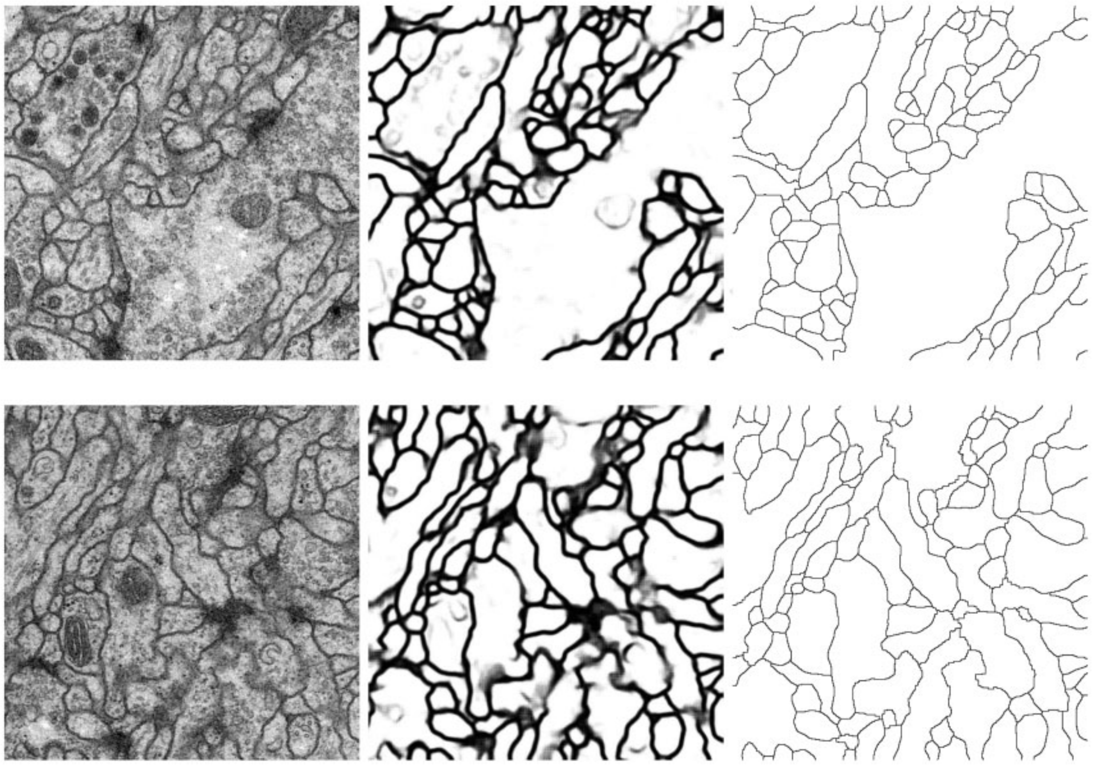

Fig. 2. Slices 1 and 12 from the testing stack are shown along with their segmentations in the top and bottom rows, respectively. The first column represents the raw input image, the second column represents the probability map output of the pixel classifier and the third column represents the final segmentation after the post-processing

Table 4. Comparison between our method and the other competing techniques

Group	Rand error [0.10^{-3}]	Warping error [0.10^{-6}]	Pixel error [0.10^{-3}]
DIVE-SCI (Our method)	17	307	58
IDSIA-SCI	18	616	102
optree-idsia	22	807	110
motif	26	426	62
SCI	28	515	63
Image Analysis Lab Freiburg	38	352	61
Connectome	45	478	62
IDSIA-V	46	462	61

In comparison to the second best approach (Ciresan *et al.*, 2012a) which was also based on a DNN and uses the same post-processing technique, our network is deeper and uses a highly optimized network architecture. The largest kernel size used in that network was 5 and the bottom layer starts with a kernel size of 4. They also implemented an actual ensemble of models where they trained several networks with different window sizes and then averaged their results. We believe that applying variations upon testing is much faster and more efficient. The larger kernels utilized in our architecture, the specific configuration of these kernels along with data augmentation techniques we applied are key factors in outperforming all the other competitive approaches in all metrics.

We demonstrate the training time requirements for the different architectures in Table 5. The comparison is based on training on 20 slices of the training stack on a GPU machine. Network C was the most computationally expensive network with the training time reaching roughly 5 days as the number of parameters to train was much higher than the other architectures. However the testing time of all networks was very fast even on our own Matlab-based CPU code with an average of few seconds on network A to roughly 4 min on network C.

Table 5. Comparison of the training time and number of parameters for different architectures

	A	B	C	D
Training hours	5	30	120	104
No. of parameters	169,580	196,100	7,615,208	5,719,016

3.1 Most recent results

Due to the ongoing nature of the ISBI 2012 challenge, the evaluation metrics along with the leading groups are continuously changing. During the review process of this work, the organizers have published new evaluation metrics, leading to new rankings in the leader board along with new participating teams. In this section, we list the new metrics along with the updated team rankings.

The challenge organizers revealed that the old metrics were not sufficiently robust to variations in the widths of neurite borders (Arganda-Carreras *et al.*, 2015). They proposed two new metrics based on specially normalized versions of the rand error and variation of information, namely foreground-restricted rand scoring after border thinning (rand score thin) and foreground-restricted information theoretic scoring after border thinning (information score

Table 6. The most recent leader board with the updated metrics that were published during the review process of this work

Group	Rand score thin	Information score thin
IAL LMC	0.9803	0.9883
IAL MC	0.9795	0.9869
CUMedVision	0.9768	0.9886
CUMedVision-motif	0.9765	0.9883
DIVE-SCI (Our method)	0.9762	0.9873
r1q	0.9747	0.9848
IDSIA-SCI	0.9730	0.9874
Image Analysis Lab Freiburg	0.9727	0.9866

We list only the top-8 groups.

thin). The latest team rankings based on these two evaluation metrics are given in Table 6.

4 Conclusion

The key contributions of our work are to provide novel insights on DNN for brain EM image segmentation and to provide record-breaking results. Our network is deep, wide and carefully designed to achieve the full performance power of DNN. We achieved the best results in the challenge without having to combine several models together, which makes the approach even more efficient. We believe that pretraining of our model on other data sets like ImageNet and combining the results of several models would further improve the accuracy. In addition, our model can generalize easily to different EM segmentation tasks due to the nature of DNNs, which learn features from data. We are also extending this deep learning scheme on the Vaa3D (Peng *et al.*, 2010) platform for other neuron images in large brain projects such as the BigNeuron (Peng *et al.*, 2015).

Acknowledgements

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research.

Funding

This work was supported in part by research grants from National Science Foundation (DBI-1147134 and DBI-1350258), Old Dominion University, and Washington State University.

Conflict of Interest: none declared.

References

Arganda-Carreras,I. *et al.* (2015) Crowdsourcing the creation of image segmentation algorithms for connectomics. *Front. Neuroanat.*, **9**, 142.

Brain Research through Advancing Innovative Neurotechnologies (BRAIN) Working Group Report to the Advisory Committee to the Director, N. (2014). BRAIN 2025, a SCIENTIFIC VISION.

Breiman,L. (2001) Random forests. *Mach. Learn.*, **45**, 5–32.

Briggman,K. *et al.* (2009). Maximin affinity learning of image segmentation. In *NIPS*, pp.1865–1873.

Briggman,K.L. *et al.* (2011) Wiring specificity in the direction-selectivity circuit of the retina. *Nature*, **471**, 183–188.

Cardona,A. *et al.* (2010) An integrated micro-and macroarchitectural analysis of the drosophila brain by computer-assisted serial section electron microscopy. *PLoS Biol.*, **8**, e1000502.

Ciresan,D. *et al.* (2012a). Deep neural networks segment neuronal membranes in electron microscopy images. In *NIPS*, pp. 2843–2851.

Ciresan,D. *et al.* (2012b). Multi-column deep neural networks for image classification. In: *CVPR*, pp. 3642–3649. IEEE.

Cireşan,D.C. *et al.* (2013). Mitosis detection in breast cancer histology images with deep neural networks. In: *MICCAI*, pp. 411–418. Springer.

Deng, J. *et al.* (2009). Imagenet: a large-scale hierarchical image database. In: *CVPR*, pages 248–255. IEEE.

Giusti,A. *et al.* (2013) Fast image scanning with deep max-pooling convolutional neural networks. *arXiv Preprint arXiv*, **1302**, 1700.

Helmstaedter,M. *et al.* (2013) Connectomic reconstruction of the inner plexiform layer in the mouse retina. *Nature*, **500**, 168–174.

ISBI. (2012). *Segmentation of Neuronal Structures in EM Stacks challenge - ISBI 2012*.

Jain,V. and Seung,S. (2009). Natural image denoising with convolutional networks. In: *NIPS*, pp. 769–776.

Jain,V. *et al.* (2007). Supervised learning of image restoration with convolutional networks. In: *ICCV*, pp. 1–8. IEEE.

Ji,S. *et al.* (2013) 3d convolutional neural networks for human action recognition. *PAMI*, **35**, 221–231.

Jia,Y. *et al.* (2014) Caffe: convolutional architecture for fast feature embedding. *arXiv Preprint arXiv*, **1408**, 5093.

Kim,J.S. *et al.* (2014) Space-time wiring specificity supports direction selectivity in the retina. *Nature*, **509**, 331–336.

Krizhevsky,A. *et al.* (2012). Imagenet classification with deep convolutional neural networks. In:7 *NIPS*, pp. 1097–1105.

LeCun,Y. *et al.* (1989) Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, **1**, 541–551.

LeCun,Y. *et al.* (2004). Learning methods for generic object recognition with invariance to pose and lighting. In: *CVPR*, vol. **2**, pp. II–97. IEEE.

LeCun,Y.A. *et al.* (2012). Efficient backprop. In: Montavon, G., Orr, G.B., Müller, K.-R. (eds.) *Neural Networks: Tricks of the Trade, LNCS, vol. 7700*. 2nd edn. Springer, Heidelberg, pp. 9–48.

Liu,T. *et al.* (2012). Watershed merge tree classification for electron microscopy image segmentation. In: *ICPR*, pp. 133–137. IEEE.

Liu,T. *et al.* (2013). Watershed merge forest classification for electron microscopy image stack segmentation. In: *ICCV*, vol. **2013**, pp. 4069. NIH Public Access.

Peng,H. *et al.* (2010) V3d enables real-time 3d visualization and quantitative analysis of large-scale biological image data sets. *Nat. Biotechnol.*, **28**, 348–353.

Peng,H. *et al.* (2015) Bigneuron: large-scale 3d neuron reconstruction from optical microscopy images. *Neuron* **87**, 252–256.

Ronneberger,O. *et al.* (2015). U-net: Convolutional networks for biomedical image segmentation. In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015*, pp 234–241. Springer.

Simonyan, K., and Zisserman, A. (2014) Very deep convolutional networks for large-scale image recognition. *arXiv Preprint arXiv*, **1409**, 1556.

Szegedy, C. *et al.* (2014) Going deeper with convolutions. *arXiv Preprint arXiv*, **1409**, 4842.

Turaga,S.C. *et al.* (2010) Convolutional networks can learn to generate affinity graphs for image segmentation. *Neural Comput.*, **22**, 511–538.

Zeiler,M.D. and Fergus,R. (2014). Visualizing and understanding convolutional networks. In: *ECCV*, pp. 818–833. Springer.

Zhang,W. *et al.* (2015) Deep convolutional neural networks for multi-modality isointense infant brain image segmentation. *NeuroImage*, **108**, 214–224.