Skeletal Muscle Cell Segmentation Using Distributed Convolutional Neural Network

Fuyong Xing
the ECE department
University of Florida
Email: f.xing@ufl.edu

Manish Sapkota
the ECE department
University of Florida
Email: msa235@ufl.edu

Fujun Liu
the ECE department
University of Florida
Email: fujunliu@ufl.edu

***Abstract*—Morphological characteristics of muscle cells, such as cross sectional areas (CSAs), are critical factors to determine the muscle health. Automatic muscle fiber segmentation is often the first prerequisite. However, it is challenging for many traditional algorithms to achieve an effective and efficient skeletal muscle cell segmentation on Hematoxylin and Eosin (H&E) stained muscle images due to the complex nature of medical imaging. In this report, we propose to formulate the cell segmentation as a pixel-wise classification problem and train a deep convolutional neural network (CNN) to segment out the cell boundaries. Considering the speed, we are implementing the CNN model in a distributed mechanism using multiple machines. We have presented the segmentation results on a set of 120 H& E stained muscle images (which produces millions of testing image patches) using the trained CNN model, and will evaluate the proposed framework with accuracy calculation and speed performance.**

I. Introduction

Skeletal muscle is one of the major tissues in human body, accounting for about 40% of body mass [1]. Muscle research community has agreed that cross sectional areas (CSAs) play an important role in determining the health and functionality of the muscle, and attempted to accurately compute the CSAs for further analysis. However, manual assessment of the CSAs is labour-intensive and suffers from inter-observer variations. Computer-aided image analysis can significantly improve the objectivity, and automatic muscle cell segmentation usually is the first prerequisite for computer-aided CSA calculation. Due to the complex nature of muscle images, there exist several major challenges for automated muscle cell segmentation (see Figure 1): 1) almost all muscle cells touch with one another and exhibit large scale variations; 2) the intensities of cell boundaries vary significantly; 3) freeze artifacts introduced during sample preparation often create false edges inside the muscle cells [2].

Automatic muscle cell segmentation is achieved by labeling the cell boundaries on digitized skeletal muscle specimens, as shown in right column of Figure 1. It has attracted a great deal of research interests in the medical image analysis community. Sertel *et al.* [3] have applied ridge detection to enhance cell boundaries and then employed morphological operations for postprocessing, but it heavily relies on the accuracy of ridge detection. In [4], a concave point based contour splitting algorithm is exploited to decompose muscle cell clumps which exhibit weak boundaries. However, local minimum along the contours often create false concave points, and therefore the subsequent contour decomposition may not be reliable. Recently, dynamic programming is employed to obtain muscle cell segmentation in [2]. It begins with generating a set of segmentation candidates and then selects a subset of region candidates using an Integer Linear Programming scheme. Several automatic cell/nuclei segmentation algorithms including Multireference level set [5], color-texture learning [6], hierarchical partial matching [7], and shape prior-constrained deformable model [8] on other image modalities can also be applied to muscle cell segmentation. However, these methods might be computationally inefficient or require sophisticated image representation design.

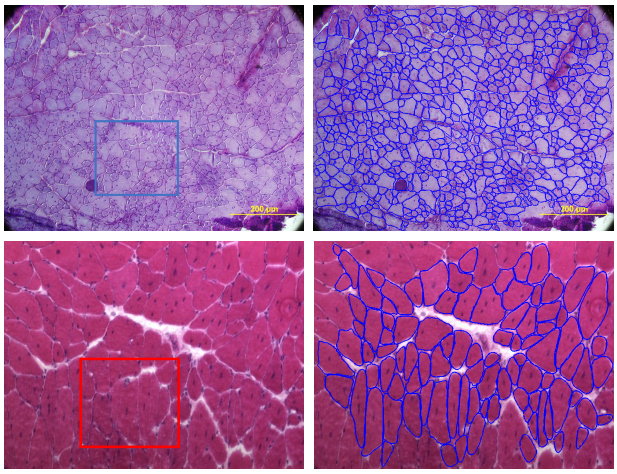

Fig. 1. The challenges for muscle cell segmentation. **Top:** Touching muscle cells with scale variations, and corresponding desired cell segmentation results. **Bottom:** Muscle cells with weak boundaries, and corresponding desired cell segmentation results. Note that cell touching image boundaries are ignored.

Recently there is an encouraging evidence that learned representation of biomedical images might perform better than the handcrafted features [9], [10], [11], and this has boosted the usage of deep learning techniques. Cruz-Roa *et al.* [12] have proposed a deep neural network for automated basal cell carcinoma cancer detection, and a unified deep representation learning model is reported [13] for automatic prostate magnetic resonance image segmentation. A deep convolutional neural network [14] has been successfully applied to

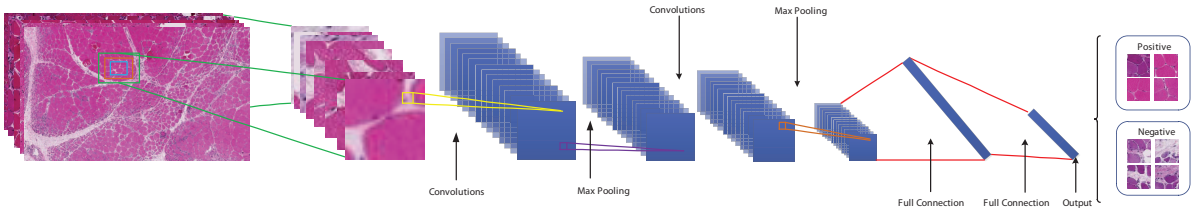

Fig. 2. The architecture of one example CNN model used for muscle cell segmentation. For illustration, we only show 7 layers here.

mitosis detection in breast cancer histopathology images, and a similar neural network [15] has been applied to membrane segmentation in electron microscopy images. However, none of these methods deal with digitized muscle specimens, which are significantly different from other types of histopathology images. Since these exist weak cell boundaries and freeze artifacts, it is very challenging to achieve automatic accurate muscle cell segmentation.

Due to the increasing of medical data, many computer-aided image analyses including cell segmentation have been applied to high-performance computing machines. Foran *et al.* have successfully exploited a grid technology called CaGrid to tissue microarray image analysis and achieve significant speed improvement. Qi *et al.* [16] have applied to multiple GPUs to cell detection to reduce running time. In addition, the robustness analysis of medical images [17] has been carried out using the CometCloud parallel computing platform, and a similar framework is presented in [18] for histopathology image retrieval.

In this paper, we present an automated cell segmentation approach on skeletal muscle images, which is based on a deep convolutional neural network (CNN). The problem is formulated into a pixel-wise classification framework, where a CNN model is trained with raw RGB values of image data and automatically learns a set of hierarchical features for classification. In the testing stage, the learned CNN model will be applied to the images in a sliding window, differentiating pixels in the cell boundaries from other regions (inside cells) to achieve automatic segmentation. We have presented the automatic cell segmentation results based on the CNN model. To improve the running time, we will perform the model training and testing in a distributed framework using multiple machines. This approach will provide efficient and effective muscle cell segmentation results, which can serve as a basis for further image analysis, such as CSA compuation, of skeletal muscle disease.

II. Cell Segmentation Using Deep Convolutional Neural Network

Given a set of training RGB image patches $I_i \in R^{r\times c\times 3}, i = 1, ..., N$ with dimensionality $r \times c$ for each of the 3 channels, we propose to learn a CNN-based mapping function to predict the class labels. The patches with center pixels located in the cell boundaries will be labeled as positive, otherwise negative.

TABLE I
The structure of the CNN used in our algorithm.

Layer No.	Layer Type	Feature Map	Kernel Size
1	Input	51 × 51 ×3	-
2	Convolutional	48 × 48 × 20	4 × 4
3	Max-pooling	24 × 24 × 20	2 × 2
4	Convolutional	22 × 22 × 20	3 × 3
5	Max-pooling	11 × 11 × 20	2 × 2
6	Convolutional	10 × 10 × 20	2 × 2
7	Max-pooling	5 × 5 × 20	2 × 2
8	Fully-connected	500 × 1	-
9	Fully-connected	250 × 1	-
10	Output	2 × 1	-

A. CNN Architecture

Convolutional neural network (CNN) is a feed-forward network composed of alternating layers of convolution and max-pooling, followed by several fully connected layers [19]. It can provide progressively abstract representation of the input with the increment of the number of layers. The architecture of one example CNN model used for skeletal muscle cell segmentation is displayed in Figure 2. The convolutional layer calculates a set of output feature maps by applying multiple kernels (filters) to the input image or feature map. Define M_j^l as the j-th output feature map of the l-th layer, we have the following equation

$$M_j^l = f(\sum_i M_i^{l-1} * K_{ij}^l + b_j^l), \quad (1)$$

where K_{ij}^l and b_j^l represent convolutional kernel and bias corresponding to the i-th input feature map and the j-th output feature map, respectively. The $f(x)$ is a nonlinear activation function, referred to as rectified linear units (ReLUs) [20]: $f(x) = \max(0, x)$. It enables fast model training and potentially improves the classification performance.

Max pooling layer is used to preform dimension reduction, and also introduces local shift and translation invariance. It corresponds to a kernel of a certain size with or without overlapping. Fully-connected layer consists of ReLUs aiming to learn global feature representation. The last (Output) layer is a fully-connected layer with a softmax function, which is used for final classification. The details of the neural network layers are summarized in Table I.

B. CNN Model Training and Testing

The model is trained using backpropagation with stochastic gradient descent [21], which locally minimizes the negative log-likelihood objective function. In order to achieve fast convergence in training, all the image patches are normalized to have zero mean and unit variance. The learning rate is an important parameter in our model. It is initialized as 0.1 and decayed by a factor of $(1 + d \times t)$ within each epoch, where d is equal to 10^{-3} and t is the epoch index, until the validation error stops improving with the current learning rate. This early stopping strategy is an important step to avoid over-fitting [22]. Batch size and the momentum are kept fixed during the training, as 100 and 0.5, respectively. In total two millions of image patches with size 51×51, half positive and half negative, are randomly generated from 60 images to train the CNN model with the open source code Cafe [23]. Fast image scanning algorithm [24] is implemented for whole image edge prediction.

In the testing stage, automatic annotation is achieved by applying the CNN model to new images using a sliding window of 51×51. The patches are normalized in a similar way as training. The patches partially outside the image boundaries are ignored. The softmax layer outputs the probabilities that each pixel is located in the cell boundaries or other regions. We predict patch labels by choosing the category associated with a higher probability.

C. Distributed Deep Learning Framework

Deep Neural Network is trained with large-scale datasets to learn millions or billions of parameters. Therefore, it is inefficient to learn these large neural networks using CPUs, and multithread programming might not improve the performance because of high data transfer latency. With the recent advancement in the GPU based computation, a significant speed up has been achieved in training. However, it requires a great deal of efforts to perform efficient GPU-based programming. Alternatively, distributed processing and computation of large-scale data, utilizing computing powers in clusters has gained momentum in recent years. Recently Dean *et al.* [25] have proposed a software framework, *DistBelief*, to train large scaled deep neural networks in clusterw. Specifically, two different algorithms have been developed within the framework: 1) an asynchronous stochastic gradient descent (named as Downpour SGD) that can beused to support network model distribution, and 2) a sandblaster L-BFGS framework that supports data and model parallelism to distribute the batch optimization procedures. Both methods allow to solve a single optimization problem distributed across multiple machines. The DistBelief framework handles communication, synchronization, and data transfer between machines during training and testing. We follow this publication and are working on the implementation of these algorithms, which are briefly described as follows.

Traditional SGD is inherently sequential which makes it unsuitable to be used for large scale optimization, the Downpour SGD is an asynchronous SGD where single model is replicated across the cluster. To avoid replicated computation across many machines, the training data are divided into different subsets and utilized to update only a portion of the model parameters. A centralized parameter server is responsible to keep track of the states of the parameters for distributed model. The updated model parameters and new computed parameter gradients are shared between machines and the centralized server via message passing. The parameter server exploits the new gradient values to update the model parameters.

The sandblaster L-BFGS framework addresses the problem of distributed batch optimization. Similar to Downpour SGD, replicas of model are created across multiple machines in the cluster, and the training data are divided into a number of subsets and each model runs on a subset of the data. Unlike Downpour SGD, the gradients for all the parameters of the model residing in one machine are computed based on the batch of data, and new computed gradients are communicated back to the parameter server to update the model.

III. Experiments

A. Data Collection

We collect the skeletal muscle images from our collaborators at University of Kentucky. The dataset consists of images with both normal and disease muscle cells, and three subtypes of muscle diseases are dermatomyositis (DM), inclusion-body myositis (IBM), and polymyositis (PM). 120 images are uniformly cropped from over 10 whole-slide scan muscle specimens, which are captured at $20\times$ magnification. Each type of muscle images has approximately the same data size. Since the CNN model is trained using small patches sampling from those images, in total we will have millions of image patches for training and testing.

B. Evaluation of Model Effectiveness

In order to test the effectiveness of the proposed CNN model, we first performs the training and testing on a single node. Figure 3 shows the cell segmentation results on eight muscle images. As one can tell, the CNN-based algorithm can produce very impressive performance, and it can provide promising results on those weak cell boundaries. To quantitatively analyze the pixel-wise segmentation accuracy, we calculate the precision P, recall R, and F_1-score as follows

$$P = \frac{|S \cap G|}{|S|},\ R = \frac{|S \cap G|}{|G|},\ F_1 = \frac{2 * P * R}{P + R}, \qquad (2)$$

where S denotes the segmentation result and G is the ground truth. The statistics of the metrics are $0.95 \pm 0.04,\ 0.77 \pm 0.14$, and 0.84 ± 0.11 for precision, recall, and F_1-score, respectively. The current CNN model takes around 30 seconds to segment one image with size around 1000×1000. We will further improve the model training by considering multiple scales such that it can capture more richer spatial information of the boundaries. We will perform the cell segmentation on the proposed distributed framework.

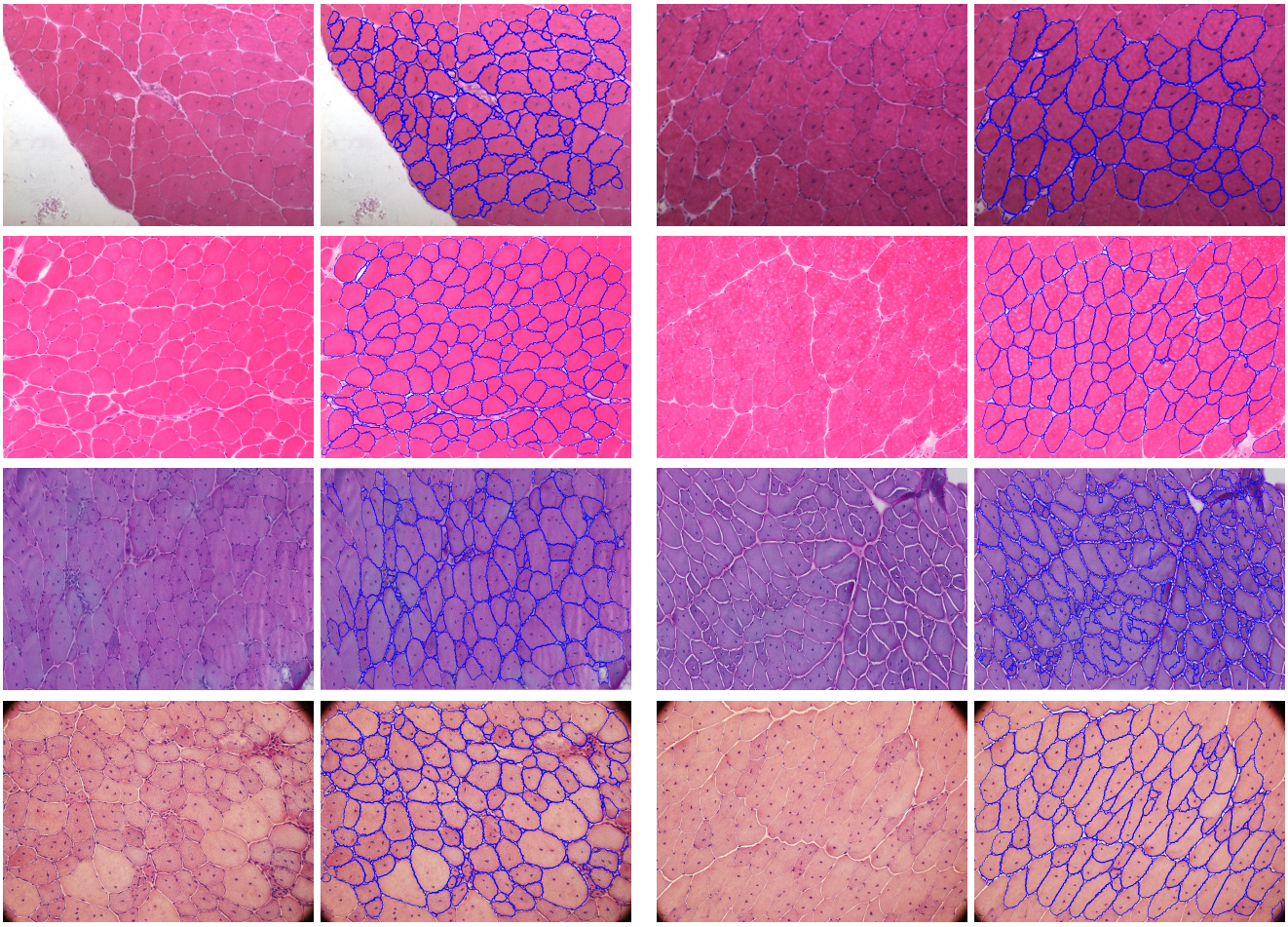

Fig. 3. Cell segmentation results on 8 muscle images using the proposed CNN model. Columns 1 and 3 denote the original images, and columns 2 and 4 represent the corresponding desired cell segmentation results. Note that cell touching image boundaries are ignored.

IV. Future Work

Our objective it to deliver a distributed CNN model for effective and efficient muscle cell segmentation. We plan to perform the CNN model training and testing in the distributed framework. To make a comparison, we will train different CNN architectures and apply them to the same data set to analyze the performance variation. This will help us better understand on the deep neural network learning on H&E stained images, especially for digitized skeletal muscle images. In addition, we will measure the running time of the proposed algorithm with respect to different worker sizes. We will analyze the speed improvement using distributed deep learning on multiple machines as well.

References

[1] F. Liu, A. L. Mackey, R. Srikuea, K. A. Esser, and L. Yang, "Automated image segmentation of haematoxylin and eosin stained skeletal muscle cross-sections," *Journal of Microscopy*, vol. 252, no. 3, pp. 275–285, 2013.

[2] F. Liu, F. Xing, and L. Yang, "Robust muscle cell segmentation using region selection with dynamic programming," in *Biomedical Imaging (ISBI), 2014 IEEE 11th International Symposium on*, 2014, pp. 521–524.

[3] O. Sertel, B. Dogdas, C. S. Chiu, and M. N. Gurcan, "Microscopic image analysis for quantitative characterization of muscle fiber type composition," *Computerized Medical Imaging and Graphics*, vol. 35, no. 7, pp. 616–628, 2011.

[4] T. Janssens, L. Antanas, S. Derde, I. Vanhorebeek, G. Van den Berghe, and F. Güiza Grandas, "Charisma: An integrated approach to automatic h&e-stained skeletal muscle cell segmentation using supervised learning and novel robust clump splitting," *Medical image analysis*, vol. 17, no. 8, pp. 1206–1219, 2013.

[5] H. Chang, J. Han, P. T. Spellman, and B. Parvin, "Multireference level set for the characterization of nuclear morphology in glioblastoma multiforme," *IEEE Trans. Biomed. Eng. (TBME)*, vol. 59, no. 12, pp. 3460–3467, 2012.

[6] H. Kong, M. Gurcan, and K. Belkacem-Boussaid, "Partitioning histopathological images: an integrated framework for supervised color-texture segmentation and cell splitting," *IEEE Trans. Med. Imaging (TMI)*, vol. 30, no. 9, pp. 1661–1677, 2011.

[7] Z. Wu, D. Gurari, J. Y. Wong, and M. Betke, "Hierarchical partial matching and segmentation of interacting cells," in *Int. Conf. Med. Image Comput. Comput. Assist. Intervent. (MICCAI)*, vol. 7510, 2012, pp. 389–396.

[8] F. Xing and L. Yang, "Robust selection-based sparse shape model for lung cancer image segmentation," in *MICCAI*, vol. 8151, 2013, pp. 404–412.

[9] G. Wu, M. Kim, Q. Wang, Y. Gao, S. Liao, and D. Shen, "Unsupervised deep feature learning for deformable registration of mr brain images," in *MICCAI*, vol. 8150, 2013, pp. 649–656.

[10] A. Prasoon, K. Petersen, C. Igel, F. Lauze, E. Dam, and M. Nielsen, "Deep feature learning for knee cartilage segmentation using a triplanar

convolutional neural network," in *MICCAI*, vol. 8150, 2013, pp. 246–253.

[11] H. R. Roth, L. Lu, A. Seff, K. M. Cherry, J. Hoffman, S. Wang, J. Liu, E. Turkbey, and R. M. Summers, "A new 2.5 d representation for lymph node detection using random sets of deep convolutional neural network observations," in *MICCAI*, 2014, pp. 520–527.

[12] A. A. Cruz-Roa, J. E. A. Ovalle, A. Madabhushi, and F. A. G. Osorio, "A deep learning architecture for image representation, visual interpretability and automated basal-cell carcinoma cancer detection," in *MICCAI*, 2013, pp. 403–410.

[13] S. Liao, Y. Gao, A. Oto, and D. Shen, "Representation learning: A unified deep learning framework for automatic prostate mr segmentation," in *MICCAI*, vol. 8150, 2013, pp. 254–261.

[14] D. C. Cireşan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Mitosis detection in breast cancer histology images with deep neural networks," in *MICCAI*, 2013, pp. 411–418.

[15] D. C. Ciresan, L. M. Gambardella, A. Giusti, and J. Schmidhuber, "Deep neural networks segment neuronal membranes in electron microscopy images," in *NIPS*, 2012, pp. 2852–2860.

[16] X. Qi, F. Xing, D. J. Foran, and L. Yang, "Gpu enabled parallel touching cell segmentation using mean shift based seed detection and repulsive level set," in *High Performance Computing (HP) workshop associated with MICCAI*, 2010.

[17] X. Qi, H. Kim, F. Xing, D. J. Foran, and L. Yang, "The analysis of image feature robustness using cometcloud," *Journal of Pathology Informatics*, vol. 3, no. 33, pp. 1–13, 2012.

[18] X. Qi, D. Wang, I. Rodero, J. Diaz-Montes, R. Gensure, F. Xing, H. Zhong, L. Goodell, M. Parashar, D. Foran, and L. Yang, "Content-based histopathology image retrieval using cometcloud," *BMC Bioinformatics*, vol. 15, no. 1, p. 287, 2014.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.

[20] V. Nair and G. Hinton, "Rectied linear units improve restricted boltzmann machines," in *ICML*, 2010, pp. 807–814.

[21] Y. A. LeCun, L. Bottou, G. B. Orr, and K. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*, vol. 1524, 1998, pp. 9–50.

[22] Y. Bengio, "Learning deep architectures for AI," *Foundations and trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[23] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the ACM International Conference on Multimedia*. ACM, 2014, pp. 675–678.

[24] A. Giusti, D. C. Cireşan, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Fast image scanning with deep max-pooling convolutional neural networks," *arXiv preprint arXiv:1302.1700*, 2013.

[25] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le *et al.*, "Large scale distributed deep networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1223–1231.