VOLUME SEGMENTATION USING CONVOLUTIONAL NEURAL NETWORKS WITH LIMITED TRAINING DATA

Hsueh-Chien Cheng *Amitabh Varshney*
Department of Computer Science and UMIACS, University of Maryland
College Park, Maryland, USA

ABSTRACT

Much of the success of convolutional neural networks (CNNs) is due to the enormous collections of labeled data, powerful GPUs, and modern network architectures that facilitate the training and testing of larger and deeper models. The limited size of labeled volumetric microscopy images, however, hinders the proper training of CNNs for volume segmentation. Here, we design various 2D and 3D CNNs with factorized convolutions and online feature-level augmentations to address challenges due to the scarcity of training data. Based on the experimental results, we found that the 3D CNNs consistently outperformed the 2D counterparts. In addition, the 3D CNN that uses both factorized convolutions and online feature-level augmentations achieved the best segmentation performance.

Index Terms— Convolutional neural networks, volume segmentation, microscopy images

1. INTRODUCTION

Recently convolutional neural networks (CNNs) have been successfully applied to segment natural image datasets such as the Pascal Visual Object Classes (VOC) dataset [1]. Typical training of a CNN requires many labeled samples (e.g. 6929 labeled images in Pascal VOC 2012) that cover a wide variety of the target objects. Nevertheless, microscopy image datasets contain considerably fewer labeled samples, usually no more than a few hundred. The limited availability of data arises from high costs in collecting and labeling microscopy images. Whereas deeper CNNs have achieved better performance in the past [2], the scarce training data increasingly raises the risk of overfitting as model complexity grows.

Data augmentations directly address the scarcity of training samples by generating additional similar samples from the existing ones. Standard augmentation operations such as random rotation, scaling, and cropping have been shown to improve prediction accuracy for image classification [3]. Nevertheless, these operations implicitly assume specific properties (e.g. rotation- and scale-invariance) of the target objects in the data; these assumptions are not always realistic or valid in applications other than image classification. To the best of our knowledge, there has been no systematic evaluation on data augmentations for microscopy data segmentation.

Besides using the application-dependent data augmentations, we approach the problem by finding suitable CNN architectures for volumetric segmentation in general. Both 2D networks, which treat the volume as a stack of image slices and segment slice-by-slice [4], and 3D networks that directly describe the spatial relationships in all three spatial dimensions reported promising results in the past. One severe drawback of 2D networks is that they do not exploit information across slices. On the other hand, the power of 3D convolutions [5] comes at the expense of a significant increase in the number of parameters, which may result in overly complicated networks. A factorized CNN uses only low-rank filters instead of full-rank ones, thus leading to a simplified model that reduces computation, avoids overfitting, and improves generalization for image classification problems [6]. Although factorizations appear to be powerful for applications with scarce training data, their effectiveness in volumetric segmentation is yet to be studied.

Given the strengths and drawbacks for both 2D and 3D networks, in this paper we study how they compare with each other empirically in terms of segmentation performance for microscopy datasets. We design and evaluate the performance of 2D and 3D CNNs (Section 2.1) that use factorized convolutions (Section 2.2) and online feature-level augmentations (Section 2.3). We modify the shortcut connection in residual blocks [2, 7] to perform online feature-level augmentations by combining features sampled from a controlled random neighborhood. Finally, we train the CNNs with a Jaccard index-based loss function (Section 2.4) to alleviate the class imbalance problem, which is also aggravated by the limited training data. Our experimental results show that the factorized 3D CNN with online feature-level augmentations performed the best among nine variants of CNN architectures, including the widely used U-Net [8].

2. CONVOLUTIONAL NEURAL NETWORKS

Our architecture of the CNNs follows the fully convolutional network [9] that takes an input image and outputs the corresponding dense prediction, of the same spatial resolution as the input image. For simplicity, we describe only the 2D CNNs here; for 3D CNNs, replace the relevant components by the 3D counterparts (e.g. 2D by 3D convolutions).

2.1. Network architecture

The CNN contains the encoding and the decoding part that gradually decreases and increases spatial resolution, respectively (Fig. 1). The decoding part, which is shown to be less influential to the overall segmentation performance [10], has much fewer parameters than the encoding part. The CNN consequently has an asymmetric structure as opposed to the symmetric U-Net [8]. We use batch normalization (BN) [11], Parametric Rectified Linear Unit (PReLU) [12], and dropout [13] in the building blocks shown in Fig. 1.

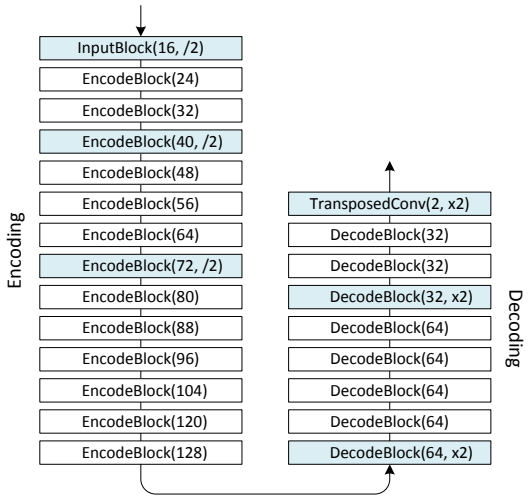

Fig. 1. The CNN contains the encoding part and the decoding part. The numbers enclosed in the parentheses represent the numbers of output feature maps. A "/2" or "x2" indicates a change in spatial resolution after the (blue) block.

Given the input images, we first concatenate the result of max-pooling with the result of stride-two convolution in the input block. The following encoding blocks are implemented as pre-activate residual blocks [7]. In each block, we stack two 3×3 convolutional layers to enlarge the receptive field with a manageable increase in the number of parameters [14]. The number of feature maps is increased by eight after each encoding block (left column of Fig. 1) to avoid a sudden increase in the number of feature maps [15]. We use the same zero-padding strategy along the feature dimension [15] for the shortcut connections when the number of feature maps changes (white encoding blocks in Fig. 1). When spatial resolution decreases (blue encoding blocks in Fig. 1), the shortcut connections combine features after applying downsampling and projection. We will elaborate more on modifying the downsampling operation to perform feature-level augmentations later in Section 2.3.

In the decoding part, we use the conventional bottleneck blocks [7], which perform convolutions in a reduced intermediate space created by projection, to further reduce the number of parameters. When spatial resolution increases (blue decoding blocks in Fig. 1), we replace the convolution by a stride-two 3×3 transposed convolution [9] to double the spatial resolution.

2.2. Factorized convolutions

Although 3D convolutions are powerful in describing the relationships among objects along the z-axis, they use 3D kernels that contain an order of magnitude more parameters than 2D kernels. For example a 3×3 kernel has nine weights whereas a $3 \times 3 \times 3$ kernel has 27 weights. The significant amount of parameters in 3D CNNs may cause the model to overfit the training data and degrade its performance during testing. When only given a limited amount of training data, the overfitting problem is a major concern, especially for the 3D CNNs.

Factorized convolutions that approximate a full-rank 2D kernel with multiple low-rank kernels have been shown to reduce computation without compromising performance [6]. Using low-rank kernels also reduces the number of parameters. For example, the total number of parameters decreases from nine to six by approximating a 3×3 full-rank kernel by a 1×3 and a 3×1 rank-1 kernel. Here we replace the m feature maps generated by the 3×3 full-rank convolution in an encoding block by the concatenation of two rank-1 convolutions, each outputs $\frac{m}{2}$ feature maps. The same strategy is applied to the full-rank 3D kernels.

2.3. Augmentations

Data augmentation is a common practice to enlarge the training set. Assuming that the target structures are rotation-invariant, we flip the image along the x- and y-axes (and z-axis too in the 3D CNNs) to create more training samples. We also apply elastic deformation [8], which is a commonly used deformation technique for creating realistic tissue images with variations.

Besides the aforementioned data-level augmentations, we modify the residual block to perform online augmentation at the feature-level when downsampling. This idea is motivated by the fractional max-pooling [16], which creates a random spatial sampling pattern to allow non-integer strides when pooling. Because the sampling pattern varies, the same input creates a large set of candidate results, perturbing the results in all subsequent layers. Here we apply a similar idea and implement the downsampling used in shortcut connections as a random spatial subsampling (Fig. 2). Whereas the original design adds the input feature maps at specific locations (e.g. all odd rows and odd columns when stride equals two) to the results of convolution [2], we apply a sampling strategy to allow the flexibility of selecting from a local neighborhood. This flexibility introduces augmentations at the feature level and can be used together with other data-level augmentations.

The stochastic sampling pattern is generated by first partitioning each spatial dimension of the input feature maps into

non-overlapping regions of size t and then selecting $\frac{t}{s}$ samples randomly inside of each region (bottom right of Fig. 2). The samples selected for all dimensions are combined into a complete sampling pattern that outputs feature maps with spatial resolutions identical to that created by subsampling with stride s (top right of Fig. 2). In fact, subsampling can be regarded as a special case of the stochastic downsampling. The parameter t controls the degree of randomness in the sampling pattern; a larger t leads to a more aggressive augmentation. In this work we choose $t = 4$. During testing, we replace this stochastic operation by an average pooling of the same stride s for a deterministic behavior.

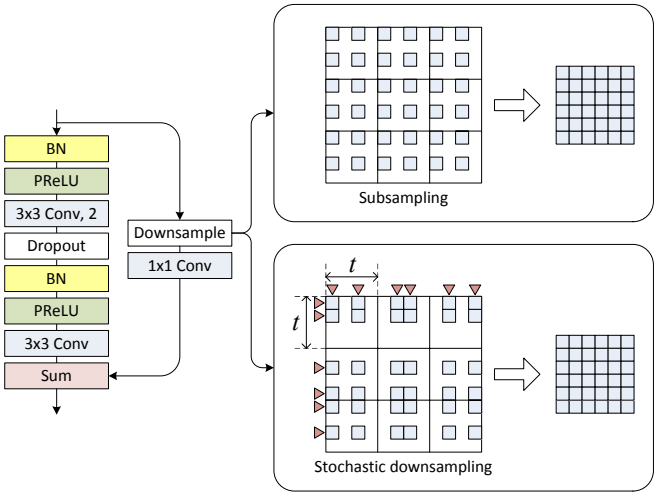

Fig. 2. We replace the subsampling (top-right) by the stochastic downsampling (bottom-right) when the encoding block reduces spatial resolution. This stochastic operation introduces augmentations at the feature level inside of a residual block.

2.4. Jaccard index-based loss

Class imbalance is a severe problem in training with limited microscopy images. Ordinary loss functions, such as the binary cross-entropy loss, usually end up emphasizing too much on the majority classes while ignoring the minority classes. Although training with balanced samples followed by post-processing the prediction probabilities during testing reported promising results [17], this procedure does not exploit the actual class distributions during training and relies on an ad-hoc postprocessing to readjust the probabilities.

In this work, we use an alternative loss function based on Jaccard index, which is also used to measure segmentation performance. For a class i, the Jaccard index-based loss l_{Jacc} is calculated as:

$$l_{\text{Jacc}}(p, y, i) = 1.0 - \frac{\sum_j p_{ij} \cdot y_{ij} + C}{\sum_j p_{ij} + \sum_j y_{ij} - \sum_j p_{ij} \cdot y_{ij} + C}$$

where p_{ij} denotes the predicted probability of the j-th voxel belonging to class i, and y_{ij} is the one-hot encoding of the ground truth labels that equals one if and only if the j-th voxel belongs to class i. The constant C, which we set to 1.0 in this work, represents a smoothing term and prevents dividing by zero. Because our target segmentation problem is a binary one, we use only the l_{Jacc} of the minority class (i.e. $l_{\text{Jacc}}(p, y, 1)$ assuming that the minority class is labeled as one) as the final loss function. The average l_{Jacc} over all classes can be used in cases of multiclass segmentation.

3. EXPERIMENTS AND RESULTS

We evaluate our CNNs with the mitochondria dataset [18]; this dataset contains two fully-labeled volumes, each of size $1024 \times 768 \times 165$, obtained using an electron microscope. The target objects in this dataset are the mitochondria, which are dark objects of oval or elongated shape. The training volume contains only 42 (complete and partial) mitochondria. The CNNs are trained to solve a binary segmentation problem (i.e. mitochondria and non-mitochondria) with a significantly imbalanced class distribution (about 5% mitochondria).

The input samples are random patches of size 256×256 (2D), extracted from the xy-plane, and $128 \times 128 \times 96$ (3D). We ensure that the patches have mitochondria voxels near the center to avoid using patches that are almost exclusively non-mitochondria. All CNNs are trained using the stochastic gradient descent solver with 0.9 momentum and L_2 regularization based on the Jaccard index-based loss. We set the batch size to 24 (2D) and three (3D) and train for 30K iterations. The learning rate starts from 0.05 (2D) and 0.1 (3D) and is multiplied by 0.1 at 50% and 75% of the iterations. The dropout rate is set to 0.1. We implement all the CNNs using Theano [19] and train them using an NVIDIA GTX1070 GPU with cuDNN library. The training took five to six hours (2D) and eight to ten hours (3D).

The results in Table 1 show the performance of U-Net [8] and eight variants of CNNs. The baseline 2D (model A) and 3D (model E) CNNs that use full-rank kernels and conventional subsampling-based shortcuts are represented by (2D, Full, –) and (3D, Full, –). The variants that use factorized convolutions and online feature-level augmentations are indicated by "Fact." and "A", respectively. We use the Jaccard index of mitochondria as the representative measure for overall segmentation quality.

U-Net is a fully convolutional network with symmetric encoding and decoding. Because of the large numbers of feature maps as well as the symmetric structure, U-Net has significantly more parameters than our baseline 2D CNN (model A). Despite using considerably fewer parameters, model A performed better than U-Net (84.4 vs. 83.5). The superior performance-per-parameter ratio of model A is a result of both the asymmetric structure that focuses on the encoding part and residual learning that uses parameters more effectively than conventional models.

The overall difference in performance between the 2D

Table 1. Segmentation results obtained by U-Net and eight network variants (A–H). The number of parameters is in millions.

Model	#Params	Precision	Recall	Jaccard
U-Net [8]	36.97M	99.3 / 94.8	**99.7** / 87.6	99.0 / 83.5
A (2D, Full, –)	1.68M	99.5 / 91.4	99.5 / 91.6	99.1 / 84.4
B (2D, Full, A)	1.68M	99.5 / 93.0	99.6 / 90.8	99.1 / 85.0
C (2D, Fact., –)	0.60M	99.6 / 91.9	99.5 / 93.1	99.2 / 86.1
D (2D, Fact., A)	0.60M	99.6 / 92.5	99.6 / 93.1	99.2 / 86.5
E (3D, Full, –)	4.94M	**99.7** / 92.5	99.6 / **93.9**	99.3 / 87.2
F (3D, Full, A)	4.94M	99.6 / 93.2	99.6 / 93.4	99.3 / 87.5
G (3D, Fact., –)	0.63M	**99.7** / 92.7	99.6 / **93.9**	99.3 / 87.4
H (3D, Fact., A)	0.63M	99.6 / **95.0**	**99.7** / 93.3	**99.4** / **88.9**

* The numbers separated by slashes correspond to the results for non-mitochondria (left) and mitochondria (right)
** The best results are marked in bold.

(A–D) and the 3D (E–H) CNNs showed that the object relationships along the z-axis are critical in volume segmentation. Even when the numbers of parameters are comparable, the 3D CNNs (G and H) performed much better than the 2D CNNs (C and D). The results suggest that the 2D CNNs are limited by the lack of contextual information across the z-axis. For example, Fig. 3 shows the segmentation results of the 80th slice. The 3D CNN (model H) predicted most accurately whereas the 2D CNNs (U-Net and model D) were unsuccessful in several cases e.g. false positive near the top-left corner (U-Net and model D) and false negative near the top-right corner (U-Net).

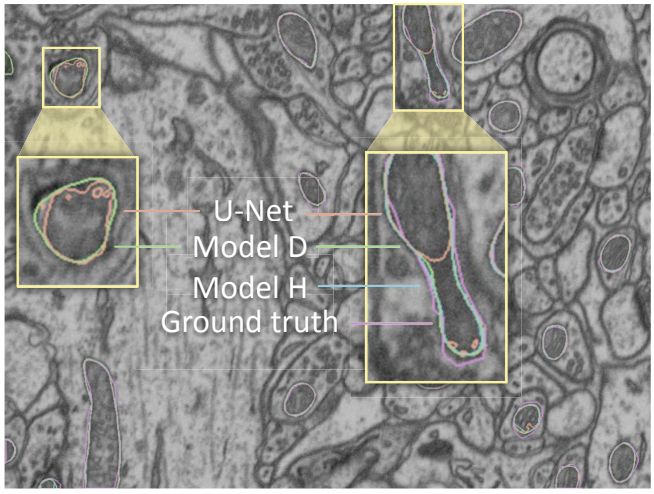

Fig. 3. This example compares the segmentation results of U-Net (orange), 2D CNN (model D, green), and 3D CNN (model H, blue), with the ground truth (purple). The 3D CNN showed promising result in which the errors are mostly near the ambiguous object boundaries.

Our observations for factorizations are consistent with the previous studies [6]: factorized CNNs offer similar (G vs. E) or better performance (C vs. A, D vs. B, and H vs. F) than the non-factorized ones. In fact, in three out of four cases factorization increased the Jaccard index of the foreground class by more than one percent. The results suggest that the factorized CNNs can still learn meaningful high-level features even though each low-rank kernel is less powerful than the full-rank one. The networks with feature-level augmentations performed either slightly better (B vs. A, D vs. C, and F vs. E) or much better (H vs. G) than the ones without such augmentations. The promising results show that feature-level augmentations in the residual blocks can introduce a suitable degree of perturbations controllable using only one parameter t.

We also assess the effect of elastic deformation by training model H without it. Surprisingly, without such deformation the Jaccard indices for both classes (99.4 and 88.8) remain competitive. Because elastic deformation can disrupt the smooth mitochondria boundaries and introduce noise in object shapes, its adverse effect may offset the benefits of augmentations. We will need a specialized augmentation procedure to achieve the full potential of data-level augmentations.

Our best segmentation result (model H) has a VOC score [1] of 94.2. A result of 94.8 is reported in [18], which combines the strengths of kernelized features, extracted by a two-stage training, and subgradient method. In contrast, our CNNs are trained end-to-end. Furthermore, errors made by a machine classifier in many cases are due to the inconsistency near the ambiguous object boundaries in the human-labeled ground truth of this dataset [20]. Fig. 3 confirms that most errors indeed happened near the boundaries.

4. CONCLUSIONS

In this work, we address the challenge of training convolutional neural networks (CNNs) with scarce training data. We compare the performance of various 2D and 3D CNNs trained with limited training samples. Based on the result of our experiments, we found that the 3D CNNs outperformed the 2D ones by a significant margin. Factorized convolutions and online feature-level augmentations improved segmentation performance the most when used together. We plan to apply semi-supervised learning and generative models to further address the problem of scarce training samples. We are also interested in applying deep learning approaches for visual knowledge discovery for biomedical datasets [21, 22].

5. ACKNOWLEDGEMENTS

This work has been supported in part by the NSF Grants 14-29404, 15-64212, the State of Maryland's MPower initiative, and the NVIDIA CUDA Center of Excellence. Any opinions, findings, conclusions, or recommendations expressed in this article are those of the authors and do not necessarily reflect the views of the research sponsors.

6. REFERENCES

[1] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

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

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proceedings of the Annual Conference on Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[4] S. Pereira, A. Pinto, V. Alves, and C. A. Silva, "Brain tumor segmentation using convolutional neural networks in MRI images," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1240–1251, May 2016.

[5] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3D convolutional networks," in *Proceedings of IEEE International Conference on Computer Vision*, 2015, pp. 4489–4497.

[6] Y. Ioannou, D. Robertson, J. Shotton, R. Cipolla, and A. Criminisi, "Training convolutional neural networks with Low-Rank Filters for Efficient Image Classification," in *Proceedings of International Conference on Learning Representations*, 2016.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proceedings of European Conference on Computer Vision*, 2016, pp. 630–645.

[8] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. 2015, pp. 234–241, Springer.

[9] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[10] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "ENet: A deep neural network architecture for real-time semantic segmentation," *arXiv preprint arXiv:1606.02147*, 2016.

[11] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of International Conference on Machine Learning*, 2015, pp. 448–456.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.

[13] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of International Conference on Learning Representations*, 2015.

[15] D. Han, J. Kim, and J. Kim, "Deep pyramidal residual networks," *arXiv preprint arXiv:1610.02915*, 2016.

[16] B. Graham, "Fractional max-pooling," *arXiv preprint arXiv:1412.6071*, 2014.

[17] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Deep neural networks segment neuronal membranes in electron microscopy images," in *Proceedings of the Annual Conference on Advances in Neural Information Processing Systems*, 2012, pp. 2843–2851.

[18] A. Lucchi, P. Márquez-Neila, C. Becker, Y. Li, K. Smith, G. Knott, and P. Fua, "Learning structured models for segmentation of 2-D and 3-D imagery," *IEEE Transactions on Medical Imaging*, vol. 34, no. 5, pp. 1096–1110, 2015.

[19] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv preprint arXiv:1605.02688*, 2016.

[20] A. Lucchi, C. Becker, P. M. Neila, and P. Fua, "Exploiting enclosing membranes and contextual cues for mitochondria segmentation," in *Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2014, pp. 65–72.

[21] C. Y. Ip, A. Varshney, and J. JaJa, "Hierarchical exploration of volumes using multilevel segmentation of the intensity-gradient histograms," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2355–2363, 2012.

[22] S. Bista, J. Zhuo, R. P. Gullapalli, and A. Varshney, "Visualization of brain microstructure through spherical harmonics illumination of high fidelity spatio-angular fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2516–2525, Dec. 2014.