A Learning-guided Hierarchical Approach for Biomedical Image Segmentation

Huaipan Jiang*, Anup Sarma*, Jihyun Ryoo*, Jagadish B. Kotra†, Meenakshi Arunachalam‡
Chita R. Das*, Mahmut T. Kandemir*
*The Pennsylvania State University, University Park, PA-16802.
Email: {hzj5142, avs6194, jihyun, das, kandemir}@cse.psu.edu
†AMD Research, Email: Jagadish.Kotra@amd.com
‡Intel, Email: meena.arunachalam@intel.com

***Abstract*—In recent years, deep convolutional networks have been widely used for a variety of visual recognition tasks, including biomedical applications. In most studies related to biomedical domain (e.g., cell tracking), the first step is to perform symmetric segmentation on target images. Such image datasets have usually the following challenges: (1) lack of human labeled training data, (2) the locations of the objects in the images are equally important as the object classification, and (3) the accuracy result is more critical than traditional image segmentation. In order to address these problems, recent studies apply large deep neural networks to perform segmentation on biomedical images. However, such neural network approaches are very compute intensive due to the high resolution and large quantity of electron microscopy data. Additionally, some of the efforts of prior neural network models are redundant since these biomedical images usually contain smaller regions of interest. In this paper, we propose a more efficient framework, especially suited for embedded computing, for image segmentation, which involves "tiling" the image first followed by processing the tiles that only contains an object of interest in a hierarchical fashion. Experimental evaluation using four datasets show that our tiling-based approach can save about 61% of execution time on average while achieving, at the same time, a slightly higher accuracy compared to the baseline approach.**

***Index Terms*—biomedical image segmentation, attention model, cell tracking**

I. Introduction

Image segmentation is a technique predominantly used in solving problems from biomedical domain, some of which include neural structure construction [1], cell tracking [2], [3], and retina layer segmentation [4]. The advent of high-throughput electron microscope makes the high resolution medical data ubiquitous, thereby posing important challenges on the compute capabilities of on-chips systems performing image segmentation.

Usually, the first step in processing these biomedical image applications is to *segment* each image into different parts. For example, in the cell tracking application, each image is segregated into foreground and background channels. While the foreground contains the cells that need to be tracked, the background information is discarded as it contains trivial information. The conventional strategies used in image segmentation include: (1) employing clustering techniques (e.g., k-means), (2) edge detection, and (3) region growing. However, these conventional approaches lack high accuracy segmentation for complex images.

Machine learning based techniques like Convolutional Neural Networks (CNN) have increased the accuracy of biomedical applications in recent past [5]–[9]. Though the CNNs provide higher accuracy in image segmentation tasks, the time spent and energy cost during the training and inference phases limits the ubiquitous adoption of these neural network based algorithms in system-on-chips. In order to address the performance inefficiency issue of these neural networks, recent studies proposed some attention models [10]–[12]. Those attention models can generate multiple small Regions of Interest(RoI) before employing the segmentation pipelines. However, those attention models are not flexible to different datasets as they can only propose fixed sizes and shapes of RoI (e.g., Faster R-CNN can generate 9 types of RoI). This limitation result in the R-CNN generating thousands of candidate RoI for each image, which in turn leads to a large mount of redundant computation.

The large mount of redundant computation limits the benefits of employing such application in system-on-chips as the time and energy cost increases. Inspired by those limitations, in this paper, we propose a *hierarchical framework* involving "tiling" mechanism, which significantly reduces the inference time and limits the amount of the redundant work. More specifically, we explore three different tiling strategies. Based on four cell tracking datasets, our best tiling strategy can reduce large portion of the computing and save the execution time on an average by 61% over the state-of-the-art segmentation framework, while maintaining the same levels of accuracy (and achieving slightly better accuracy in some cases).

This paper is organized as follows. We include a brief primer on CNN architectures in Section II-A, and then compare our proposal to related works in Section II-B. In Section III, we present our motivational results, following which we discuss our methodology in Section IV. After presenting our proposed hierarchical framework and our three tiling strategies, we give our experimental results and compare our approach quantitatively against other approach in Section V, and conclude the paper with a summary in Section VI.

Identify applicable funding agency here. If none, delete this.

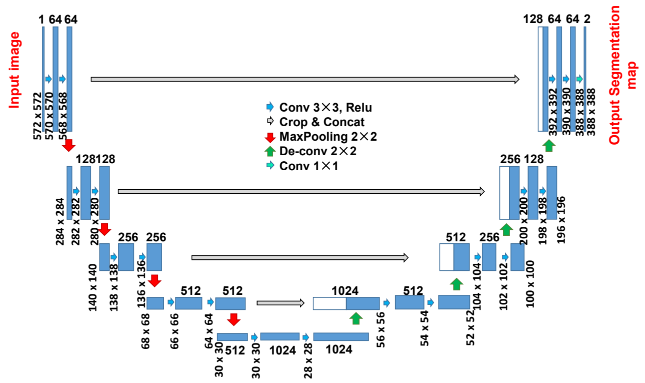


Fig. 1: U-net architecture [8]. Each box represents a feature map where the number of channels indicated on top (bottom) of each box. The height and width are specified at the lower corner of each box.

II. Background and Related Work

A. Background

Convolutional Neural Networks (CNN) have been successfully deployed in the context of image classification and segmentation problems. The classification CNNs include LeNet [13], Deepyeast [14], VGGNet [15] and AlexNet [16]. In general, a neural network assigns a single label for each image, depending on type of objects contained in the image. For example, the MNIST [17] dataset has 10 classes of labels corresponding to digits 0-9. The image segmentation networks [8], [9], [18], on the other hand, classify all the pixels in an image, and then assign a unique class ID to a group of connected pixels. For example, FCN [9] identifies objects on the image in the PASCAL VOC dataset [19].

Compared to the traditional neural networks, image segmentation networks in biomedical domain are computationally more intensive. The main reason is that this type of task typically lacks enough training data and are more sensitive to accuracy. To achieve high accuracy for image segmentation, complex network structures have been designed to maintain more weights inside the network. For example, U-net [8] (Figure 1) contains a total of 64 layers, including 19 convolution layers. It also combines the high resolution features with lower resolution features after several pooling layers.

B. Related Works

Our approach targets at reducing the execution time of the biomedical image segmentation tasks, where we need to detect all the objects of interest in a given image. Several prior works have been done towards image segmentation, e.g. Ciresan et al. [5] applied a sliding-window on the input image where each pixel in the image is given a label based on the neighboring chunk pixels. This strategy is inefficient due to the overlap areas between each windows, thus all pixels are used redundantly to train the network multiple times. To address this problem, FCN [9] proposed a more elegant network which uses traditional classification neural networks initially to track the features from the image, and then combines the features with location information in the last several layers of the network. Inspired by that, U-net [8], which is mainly designed for the biomedical image segmentation tasks, refines the FCN architecture further. An input image in U-net traverses through several convolution layers, followed by pooling layers to obtain a low resolution feature map. This low resolution feature map is then passed to the "top" by the deconvolution layers, thereby resulting in a "u-shaped" architecture (see Figure 1). As the low resolution feature from the bottom layers traverses the upper deconvolution layers, it is concatenated with the features from the corresponding max-pooling layers. This concatenation results in a high precision feature in the result. SegNet [18], similar to U-net, also employs a u-shaped architecture. Hosseini et al. [6] employed a cascaded hierarchical model in the content of logistic disjunctive normal networks. However, these prior approaches mainly focus on accuracy of segmentation over execution time, which is the essence of this work without compromising on the accuracy.

Recently, several attention models was proposed for recognizing the regions of interest inside the images. [10] uses Recurrent Neural Network (RNN) which remembers the previously captured information over time in determining the portion of the image containing the text. Following this phase, the text localization pipeline can detect the text. Besides this, there are several R-CNN (Region-based Convolutional Neural Network) based mechanism [11], [12] that generate candidate regions in an image. Each candidate region corresponds to a possible object in the image. For example, the Faster R-CNN [11] applies a sliding window on the feature map generated from the last convolution layer of VGGNet. Each sliding window is sent to a Region Proposal Network (RPN) for generating several candidate regions with different sizes and anchors. The Mask R-CNN [12] adds another branch to Faster R-CNN, which provides a segmentation mask for each detected object in the regions.

However, the R-CNN based works have the following drawbacks: (1) the redundant computation due to the overlap area in the sliding window process, (2) such RPN frameworks generate up to thousands of candidate regions for each image, (3) these frameworks are not flexible when customizing the framework of the segmentation process. This is important because different datasets are suitable for different frameworks, and (4) when the input image size is large, the memory become a bottleneck when implement R-CNN, especial for an on-chip system [20].

Hence, it is promising to propose a more efficient framework which can reduce the execution time of the inference phase of the CNN based image segmentation applications. In our approach, we first identify possible location of the objects (cells) in an image, and then only focus on the particular locations in the image, thereby improving the overall execution time. Note that, our approach is oriented towards reducing the execution time without negatively impacting accuracy.

III. Motivation

In many existing biomedical datasets [2], [3], each images have nearly one million pixels, which is significantly larger

Datasets	Pixel-wise TR	Tile-wise TR
PhC-C2DH-U373	6.47%	59.01%
Fluo-C2DL-MSC	6.05%	47.73%
Fluo-N2DH-SIM+	6.30%	65.83%
Fluo-C3DH-H157	6.83%	38.53%
Average	6.41%	52.77%

TABLE I: Pixel-wise and tile-wise tissue ratios (TR).

Datasets	Accuracy	U-net time	AlexNet time
PhC-U373	99.75%	137.28s	1.09s
Fluo-MSC	99.17%	340.33s	3.47s
Fluo-SIM+	99.27%	1088.49s	8.78s
Fluo-H157	98.97%	673.73s	7.99s
Average	99.29%	430.24s	4.04s

TABLE II: AlexNet classifier accuracy and execution time. The accuracy result (second column) indicates how many tiles the classifier accurately predicts to contain cells or not. The third column is the execution time of U-net. The fourth column is the classifier execution time.

than the images in traditional image datasets [17], [21]–[23] that contain only thousands of pixels. Additionally, these biomedical datasets lack of human labeled training data. Observing this, prior research focused on designing deeper and larger neural networks to improve the overall accuracy during inference. For example, the U-net [8] model has a total of 101MB weights. The large size of each biomedical image and the complexity of the neural network itself pose challenging performance problems to the end-users, especially when these neural networks are deployed under system-on-chip environments. Input image agnostic reduction in the size of the neural network also reduces the end-to-end inference time. However, such input agnostic model compression might also result in reduced accuracy [24]. Hence, reducing the end-to-end execution time of applications without compressing the neural network is itself an open research problem.

Contrary to the prior works in the biomedical domain, we observe that *only a small portion of an biomedical image actually contains objects of interest*. In fact, as can be observed from Table I (Pixel-wise), on average, only a small fraction (6.41%) of the pixels in a biomedical image belong to the cells, while the other pixels simply represent the background. Such a small ratio of objects of interest enables us to perform image segmentation on such useful areas while discarding the not-so interesting areas. Further, such smaller areas of objects of interest give us the opportunity of employing "approximate computing", when processing these huge biomedical datasets. In our study, we first *tile* all the images into 4-by-4 tiles. From Table I (Tile-wise), one can observe that only 52% of the tiles among four datasets contain objects of interest (cells). This suggests that, if we could get a perfect filter for these tiles, we would be able to avoid a large amount of redundant computation, thereby reducing the running time. Clearly, the challenge here is to achieve this *without negatively impacting the accuracy*.

As will be discussed in the next section, instead of using the segmentation framework indiscriminately for the entire image, we propose a "hierarchical" framework for image segmentation. In our approach, the first tier neural network performs image classification on the tiles and acts as a "filter" (also referred to as the "classifier" in the rest of this paper). The classified tiles containing valid objects (output from the first tier) are then fed to the segmentation framework, which forms the second tier of our hierarchical framework. The classification network employed in the first tier filters the tiles into two classes, viz., if the tile contains an object or not. More details of our framework are given in the next section.

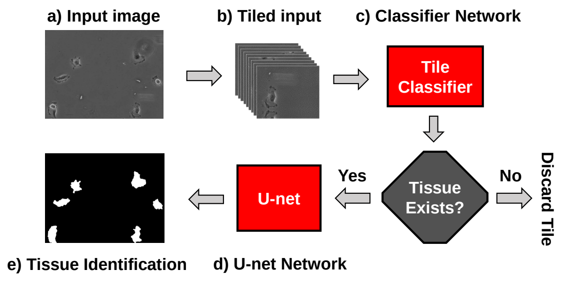


Fig. 2: High level view of our tiling-based framework.

IV. Methodology

Our framework consists of two parts. As shown in Figure 2, we have an classifier(Alexnet) on the top of the framework and a segmentation(U-net) framework at the bottom. Note that, although we use AlexNet as the classifier and U-net as the segmentation framework in our experiments, it is not necessary that we use these two networks. In fact, different types of classifier and segmentation networks can be employed, depending on the dataset characteristics.

A. *Tile Classifier and U-net Framework*

During the inference phase of our framework, the entire image is tiled by one of our tiling techniques, and each tile is then sent to the classifier network for detecting whether it contains tissue. Based on the classification result, only the tiles determined as containing tissues are forwarded to the U-net framework, thereby ensuring the image segmentation is performed only on the useful tiles.

As the classifier network determines which parts of the image needs to be segmented further by U-net, we employ a simple classification CNN for this task where this network should be accurate and not incur too much overhead (especially when a system-on-chip environment is considered). Considering the accuracy and the running time overhead, we employ AlexNet [16] as our classifier. We observed that AlexNet achieved nearly *100% of accuracy* for all the four datasets we used in our evaluations, whereas its running time was negligible compared to the U-net segmentation time (shown in Table II). This high accuracy can be attributed to the fact that the decision of whether an object exists or not is quite a simple task for AlexNet (or similar networks).

The U-net forms the heart of our hierarchical framework, and handles the image segmentation tasks for all the selected tiles coming from the classifier. In our hierarchical framework, only a subset of an image will be fed to the U-net instead of the entire image. Below, we explain the three tiling strategies we propose and evaluate in this work.

B. Fixed Tiling

The first tiling strategy we propose is *fixed tiling*. In this tiling strategy, as shown in Figure 3a, each image is divided into tiles of the same size. It is easy to see that, increasing the tile size will result in fewer tiles to be processed by the classifier (AlexNet) and can result in only a few tiles to be discarded as the likelihood of each larger tile to contain a cell increases. On the other hand, decreasing the size of a tile will result in many tiles to be discarded while increasing overall execution time which is not preferable for overall performance.

C. Adaptive Tiling

It is to be noted that, the fixed tiling strategy will face a tradeoff challenge. If the tile size is too big, then its behavior comes close to U-net. If on the other hand, the tile size is too small, the overall execution time will increase. Unfortunately, it is not trivial to easily determine an "optimal" tile size to employ, when using the fixed tiling strategy for different input datasets. Motived by this observation, we next propose *adaptive tiling*. As shown in Figure 3b, in this second tiling strategy, we first divide each image into large tiles and send those tiles to the classifier. Depending on whether the tile is detected (by the classifier) to contain a tissue or not, each of the larger tiles can be further divided into smaller tiles. These smaller tiles are sent through the classifier once again. And, this process continues in an iterative manner, where, at each step, a larger tile is further divided if it is detected to have a tissue.

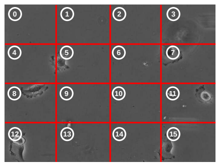
However, there is a potential issue which can limit the execution speed of the adaptive tiling strategy. The output feature map generated by U-net is smaller in size than its input size. This is because of the shrinkage during the convolution layers. For example, the 3*3 convolution kernel in U-net causes an image size loss of 2 pixels in both height and width. As shown in Figure 1, we can observe that, to obtain a segmentation map of size n*n, we need to feed U-net with a padded tile, which is of size (n+184)*(n+184). Figure 4 illustrates this issue using an example. Hence, as the number of tiles increases, each tile itself becomes smaller in size; however, the overhead of total padded area becomes relatively larger and this causes an increase in execution time due to the redundant operations on the overlap areas between the neighboring tiles. Consequently, it is not clear whether the adaptive tiling strategy could generate any better result than the fixed tiling strategy.

D. Scratch Tiling

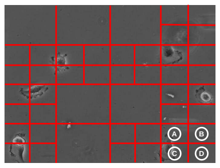
To address the above mentioned limitation for the adaptive tiling, we propose a more flexible tiling strategy that scratches the surrounding area of the objects from the image directly. We call this strategy *scratch tiling*. In this tiling strategy, we first divide the image into very fine-grained tiles and then send them to the classifier. The classifier generates a "scratch map" for each image, as shown in Figure 6. In the scratch map, an entry '1' indicates that the corresponding tile in the image contains a cell, while a '0' indicates that no cell in the tile. We can then scratch some rectangle boxes (the colored rectangles in Figure 6D) from the scratch map to cover all the 1s in the map, and then send the rectangles boxes to U-net. Unlike the fixed tiling strategy, the adjacent rectangles here can be of *different sizes* depending on the tissue sizes.

Algorithm 1 Scratching from the scratch map

```
Input: A scratch map M with size of n by m and all the entries are 0 or 1
Initialization:
Neighboring_region_list ← ∅;
for each elements $M[i, j]$ in M do
    if $M[i, j]$ is not in any connected regions now then
        call BSF start from $M[i, j]$ and get a neighboring region $R$;
        Neighboring_region_list.push_back($R$);
    end
end
$Rectangle\_boxes \leftarrow \emptyset$;
for each neighboring region R in Neighboring_region_list do
    $T1 \leftarrow$ the largest rectangle box of 1s at the up-left of R;
    $T2 \leftarrow$ the largest rectangle box of 1s at the up-right of R;
    $T3 \leftarrow$ the largest rectangle box of 1s at the down-left of R;
    $T4 \leftarrow$ the largest rectangle box of 1s at the down-right of R;
    $T5 \leftarrow$ The remaining area of 1s in R;
    $Rectangle\_boxes.push\_back$(T1);
    $Rectangle\_boxes.push\_back$(T2);
    $Rectangle\_boxes.push\_back$(T3);
    $Rectangle\_boxes.push\_back$(T4);
    $Rectangle\_boxes.push\_back$(T5);
end
Return $Rectangle\_boxes$;
```



(a) This shows the fixed tiling strategy. In this case, the image is divided into 4 by 4 tiles. Among these, tiles 0, 1, 2, 9, 10, and 13 do not contain any part of any cell, while tiles 3, 4, 5, 6, 7, 8, 11, 12, 14, and 15 contain some parts of the cells. Consequently, first group of the tiles will be omitted.



(b) This shows the adaptive tiling strategy. In this case, for those tiles which contain some parts of the cells will be further divided into 2 by 2 smaller tiles. For example, tile 15 in (a) will be divided into tiles A, B, C, and D. Since only A and C contain cells while B and D do not, tile B and tile D will be discarded.

Fig. 3: Fixed 3a and adaptive 3b tiling strategies.

To feed all the rectangle boxes containing 1 in the scratch map to U-net, we apply a scratching algorithm (presented as Algorithm 1 above). As can be observed from this algorithm, we employ Breadth First Search (BFS) to accumulate all the neighboring regions of 1s. Then, in each region, we use a greedy algorithm to scratch the rectangle boxes from each corner of the regions as large as possible. In the end, we reorganize the results of all the chunks together to obtain the final image segmentation result for the whole image, as we did in the previous two tiling strategies.

E. False Negative Correction

Although the classifier can achieve a high filter accuracy ($\sim$99%), this could lead to failure cases if we discard any tiling area containing tissues (false negative). Observing this, we also provide a correction method to avoid such false negative cases. As Figure 5 shows, if tile 1 contains cells (true positive) while tile 2 was incorrectly predicted as not having cells (false

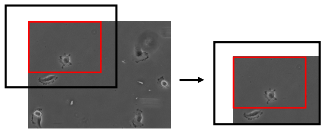


Fig. 4: To segment the tile in the red box, we feed U-net with the black box as input.

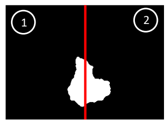


Fig. 5: Correction for False Negative issue

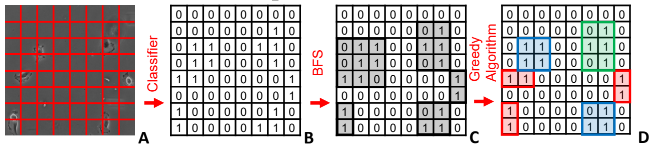


Fig. 6: This shows the scratch tiling strategy. First, the raw image (A) is divided into 8 by 8 tiles, and the resulting tiles are then sent to the classifier to generate the scratch map (B). After that, we employ BSF to accumulate all the neighboring regions as shown in (C). Finally, we use a greedy algorithm to obtain the rectangle boxes in (D).

negative). After we get the segmentation result from U-net, we can correct this mistake since some continuous pixels at the border between 1 and 2 belong to the cells. We can then resend tile 2 to the U-net, instead of discarding it. We want to emphasize that, by applying this correction strategy, we achieve a 100% classifier accuracy on our four test datasets.

V. Experimental Evaluation

We test our framework on four different biomedical image datasets [2], [3] which come from the ISBI2015 cell tracking challenge. For each dataset, we trained different models for both U-net and AlexNet. And, we report both the *speedup* and *accuracy* results with all of our three tiling strategies discussed in the previous section. The speedup results reported below represent the "normalized performance" over the baseline U-net. For the accuracy results, we use IOU (intersection over union) as our metric, which is also used in several prior studies. [2], [3], [8], [9], [18].

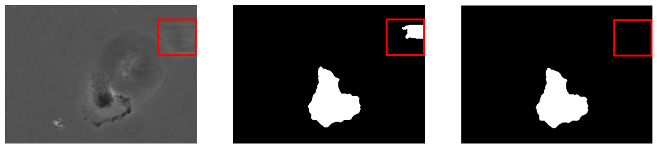
A. Datasets

We applied our framework to a cell segmentation task of the microscopic images. Our four datasets are listed below :

- **PhC-C2DH-U373 (D1):** The Glioblastoma-astrocytoma U373 cells.
- **Fluo-C2DL-MSC (D2):** Rat mesenchymal stem cells.
- **Fluo-N2DH-SIM+ (D3):** Simulated nuclei of HL60 cells which are stained with Hoescht.
- **Fluo-C3DH-H157 (D4):** GFP-transfected H157 lung cancer cells embedded in a matrigel matrix.

B. Experimental Results

To test the effectiveness of our proposed tiling strategies, we performed experiments on both CPUs and GPUs. The configurations of the architectures we used are listed in Table III. Figures 8 and 9 show the performance and accuracy results, for the fixed, adaptive and scratch tiling strategies(with different variants of fixed and adaptive tiling). As can be



(a) Raw input image. (b) Result of baseline U-net. (c) Result of tiling-based framework.

Fig. 7: False-positives in images.

CPU Configuration	
Key	Value
Number of cores	6
CPU Frequency (GHz)	2.00
Cache size (last level)(MB)	15
Memory size (GB)	64

GPU Configuration	
Key	Value
Shading Units (Cores)	2496
SMX Count	13
GPU Clock (MHz)	706
Memory size (MB)	5120

TABLE III: Configuration of Intel(R) Xeon(R) CPU E5-2620(CPU) and NVIDIA Tesla K20m(GPU).

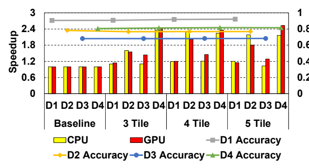
observed from these figures, all the three tiling strategies achieve better accuracy (IOU: 0.796, 0.799 and 0.799, respectively) compared to the baseline U-net (baseline) (IOU: 0.794). This is because, in the microscopy images, certain pixels can cause false-positives like bubble or halo, which are mistakenly detected as the tissues by U-net. Such false-positives can be successfully filtered by our AlexNet-based classifier network. For example, in Figure 7, the region in the upper right portion of the input image is detected as a cell by U-net, whereas, in our hierarchical framework, that object is successfully filtered by the AlexNet classifier. This is because AlexNet has a larger convolutional kernel size (up to 11*11) which can help the network cover the context information from a larger group of neighboring pixels, while U-net has a smaller convolution kernel size which extracts the context information from a smaller region.

As can be observed from Figures 8 and 9, amongst all the four datasets, the fixed tiling strategy achieves a 1.64x speedup on CPU, and a 1.70x speedup on GPU, on average; and, the adaptive size tiling strategy achieves a 1.56x speedup on CPU and a 1.65x speedup on GPU. Lastly, the scratch tiling strategy achieves 1.68x to 4.12x and an average 2.52x speedup (61% execution time reduction) which represents a significant improvement over the previous two tiling strategies.

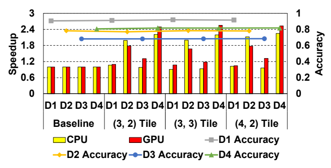
We further evaluate our scratch tiling compare with the naive U-net approach on a system-on-chip environment (Raspberry Pi 3) to show this work is suited for embedded computing. From Table IV we can observe our scratch tiling achieves a 3.03x speedup over original U-net implementation among all these four datasets in average.

C. Comparison against Faster R-CNN-based Strategies

As mentioned earlier in Section 2.2, Faster R-CNN involves redundant work when employing the RPN on the overlap area of the sliding window. Hence, the RoI proposal stage of Faster R-CNN is expected to be more time and energy consuming than our tiling approach. Figure 10 reports the performance comparison between our fixed tiling strategies and Faster R-CNN. We can observe that our fixed tiling strategy (4-by-4 tile size) achieves a better performance (3.7X in average) than Faster-RCNN. Additionally, Faster R-CNN



(a) Fixed Tiling.



(b) Adaptive Tiling.

Fig. 8: Speedup and accuracy results with the fix tiling and adaptive tiling. The bar graph shows speedup, and the line graph shows accuracy. "n Tile" indicates that the image is divided into n-by-n tiles. "(n, m) Tile" indicates that the image is first divided into n-by-n, each tile is further divided into m-by-m sub-tiles if it contains tissue.

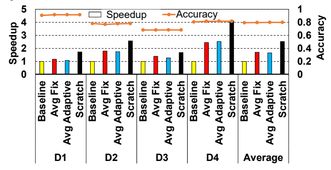


Fig. 9: Performance and accuracy results with the scratch tiling (in Tesla GPU).

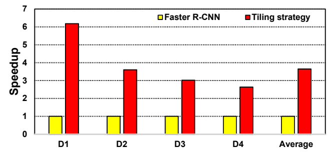


Fig. 10: Speedup of tiling-classifier over Faster R-CNN

proposes 300 candidate regions for each image. If all the candidate regions are sent to the segmentation framework, the execution time will be significantly longer since each image contains only a few tiles. Additionally, since we use tiling strategies, our images can be easily mapped to the system-on-chip environment with limited memory, whereas Faster R-CNN works on the entire large images.

VI. Conclusion

In this work, we propose a tiling-based two-level hierarchical framework for biomedical applications involving image classification and segmentation. More specifically, we propose three different tiling strategies to reduce the execution time of the segmentation framework without impacting it's accuracy. Our best performing tiling strategy, scratch tiling, improves performance by 61%, while achieving a similar accuracy as the original U-net implementation.

References

[1] A. Cardona, S. Saalfeld, S. Preibisch, B. Schmid, A. Cheng, J. Pulokas, P. Tomancak, and V. Hartenstein, "An integrated micro- and macroarchitectural analysis of the drosophila brain by computer-assisted serial section electron microscopy," *PLOS Biology*, vol. 8, pp. 1–17, 10 2010.

[2] M. Maka, V. Ulman, D. Svoboda, P. Matula, P. Matula, C. Ederra, A. Urbiola, T. Espaa, S. Venkatesan, D. M. Balak, P. Karas, T. Bolckov, M. treitov, C. Carthel, S. Coraluppi, N. Harder, K. Rohr, K. E. G. Magnusson, J. Jaldn, H. M. Blau, O. Dzyubachyk, P. Kek, G. M. Hagen, D. Pastor-Escuredo, D. Jimenez-Carretero, M. J. Ledesma-Carbayo, A. Muoz-Barrutia, E. Meijering, M. Kozubek, and C. Ortiz-de Solorzano, "A benchmark for comparison of cell tracking algorithms," *Bioinformatics*, vol. 30, no. 11, pp. 1609–1617, 2014.

[3] V. Ulman, M. Maska, K. E. G. Magnusson, O. Ronneberger, C. Haubold, N. Harder, P. Matula, P. Matula, D. Svoboda, M. Radojevic, I. Smal, K. Rohr, J. Jaldén, H. M. Blau, O. Dzyubachyk, B. Lelieveldt, P. Xiao, Y. Li, S.-Y. Cho, A. C. Dufour, J.-C. Olivo-Marin, C. C. Reyes-Aldasoro, J. A. Solis-Lemus, R. Bensch, T. Brox, J. Stegmaier, R. Mikut, S. Wolf, F. A. Hamprecht, T. Esteves, P. Quelhas, Ö. Demirel, L. Malmström, F. Jug, P. Tomancak, E. Meijering, A. Muñoz-Barrutia, M. Kozubek, and C. Ortiz-de Solorzano, "An objective comparison of cell-tracking algorithms," *Nature Methods*, pp. EP –, Oct 2017.

Dataset	U373	MSC	SIM+	H157	Average
Speedup	1.67x	2.43x	2.21x	9.35	3.03x

TABLE IV: Speedup of Scratch tiling on Raspberry Pi 3.

[4] P. P. Srinivasan, S. J. Heflin, J. A. Izatt, V. Y. Arshavsky, and S. Farsiu, "Automatic segmentation of up to ten layer boundaries in sd-oct images of the mouse retina with and without missing layers due to pathology," *Biomed. Opt. Express*, vol. 5, pp. 348–365, Feb 2014.

[5] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Deep neural networks segment neuronal membranes in electron microscopy images," in *Advances in Neural Information Processing Systems 25*, pp. 2843–2851, Curran Associates, Inc., 2012.

[6] M. Seyedhosseini, M. Sajjadi, and T. Tasdizen, "Image segmentation with cascaded hierarchical models and logistic disjunctive normal networks," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2013.

[7] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, *3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation*, pp. 424–432. Cham: Springer International Publishing, 2016.

[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*, pp. 234–241, Springer, 2015.

[9] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[10] W. Huang, D. He, X. Yang, Z. Zhou, D. Kifer, and C. L. Giles, "Detecting arbitrary oriented text in the wild with a visual attention model," in *Proceedings of the 2016 ACM on Multimedia Conference*, MM '16, (New York, NY, USA), pp. 551–555, ACM, 2016.

[11] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015.

[12] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *CoRR*, vol. abs/1703.06870, 2017.

[13] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov 1998.

[14] T. Pärnamaa and L. Parts, "Accurate classification of protein subcellular localization from high-throughput microscopy images using deep learning," *G3: Genes, Genomes, Genetics*, 2017.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, Curran Associates, Inc., 2012.

[17] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov 1998.

[18] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," 2015.

[19] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, pp. 98–136, Jan. 2015.

[20] J. B. Kotra, D. Guttman, N. C. N., M. T. Kandemir, and C. R. Das, "Quantifying the potential benefits of on-chip near-data computing in manycore processors," in *2017 IEEE 25th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, vol. 00, pp. 198–209, Sept. 2017.

[21] Y. Chong, J. Koh, H. Friesen, S. K. Duffy, M. Cox, A. Moses, J. Moffat, C. Boone, and B. Andrews, "Yeast proteome dynamics from single cell imaging and automated analysis," *Cell*, 2015.

[22] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, June 2009.

[23] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009.

[24] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," *CoRR*, vol. abs/1510.00149, 2015.