Improved On-Device Hyperdimensional-Computing-based Image Segmentation

Xinran Zhou
College of Computer and Network Security
Chengdu University of Technology
3504486686@qq.com

Huayue Lu
Department of Computer Science and Engineering
University of Connecticut
huayue.lu@uconn.edu

***Abstract*—Today's artificial intelligence technology has made great progress and in many fields involved in artificial intelligence, such as medical imaging, logistics, manufacturing, and other fields, one of the most basic tasks is segmentation. Image segmentation is a more difficult segment in segmentation, and its existing problems are: insufficient labeled data leads to weakened performance, and it is difficult to process on the device. Recently, a hyperdimensional-computing-based unsupervised segmentation approach is proposed, which can process images on-device. However, there are still some problems, like latency. In this work, I have made some improvements based on previous work and propose ImSegHDC having much lower latency with only a little segmentation performance drop.**

***Index Terms*—Unsupervised Image Segmentation, On-device Learning, Hyperdimensional Computing**

I. Introduction

In the modern era, artificial intelligence (AI) technology has reached a state of increasing maturity and finds widespread application in various fields. Edge AI, (e.g., tiny machine learning (ML)) plays an important role in these applications. For example, AI is extensively utilized in healthcare and medicine for medical imaging [1] and disease diagnosis [2]. Additionally, it is integrated into transportation for autonomous vehicles [3], traffic management systems [4], and logistics optimization [5]. In manufacturing, AI is used for quality control [6], predictive maintenance [6], and process optimization [6]. In some of these applications, e.g., medical imaging [1], one fundamental task is segmentation, which has paved the way for the emergence of AI architecture models tailored for this purpose.

Image segmentation forms a more complex task compared with the classical classification task since classification requires to classify an image to a class. In contrast, segmentation requires identifying which part of the image should be a specific target. Nowadays, deep neural networks (DNNs)-based approach dominates other approaches in the segmentation tasks. For example, the encoder-decoder-based U-Net [7] has demonstrated its great capabilities in segmentation tasks. However, there are still certain limitations when applied in practical scenarios.

The most common limitation is the requirement for substantial labeled data to train a robust network. Unfortunately, in image segmentation, obtaining sufficient labeled data for training is often a difficult and labor-intensive process, impeding the model's performance. These issues significantly affect segmentation performance, leading to potential consequences. For instance, in the medical field, inaccurate segmentation of lung delineation and lesion measurement can prevent radiologists from precisely identifying lung infections, leading to errors in analysis and diagnosis [8]. While manual labor can be utilized as an effective means to tackle these problems, it comes with staggering time and cost requirements. Thus, finding a way to achieve high-performance segmentation with limited labeled data becomes paramount. Unsupervised image segmentation could be an efficient solution as it doesn't rely on labeled data.

Additionally, data privacy protection remains a critical concern, with potential risks associated with uploading data to the cloud for processing. Ideally, data processing directly on the device would be a more desirable solution. However, in pursuit of better segmentation results, neural network models become larger and larger. This brings new challenges to putting machine learning models on edge devices. At the same time, people also need to do segmentation on edge devices to achieve more segmentation requirements.

In light of these challenges, developing unsupervised segmentation, which does not rely on labeled data for performance, emerges as a favorable option. However, unsupervised segmentation is still in its infancy, and mature frameworks are scarce. Recently, Yang et al. proposed a new framework called SegHDC [9], which successfully addressed some of the aforementioned issues. Nevertheless, it also gave rise to new challenges, such as longer module convergence time and insufficient speed when processing data directly on the device. Based on the analysis of the drawbacks, I apply some new hyperparameters to adjust the block-based Manhattan distance for position encoding. Moreover, to boost the efficiency of the edge device, I propose grouping similar colors and assigning them the same HV which enables reducing the dimensions. my main contribution is as follows:

- We adjust the block size to accommodate the non-square rectangle images.
- We bring a group factor to combine a number of colors and thus can reduce the dimension used.
- We bring a new centroids selection function to boost the cluster convergence.

The subsequent sections of this paper are organized as

follows: Section II delves into the related work, while Section III presents the SegHDC framework proposed by Yang et al. Section IV describes the specific improvement methods, Section V presents the experimental results, and finally, Section VI offers the conclusion.

II. Related work

A. Image Segmentation

Image segmentation refers to the technique and process of dividing an image into specific and unique regions, aiming to identify and propose objects of interest. It represents a crucial step in transitioning from image processing to image analysis. From a mathematical perspective, image segmentation involves dividing digital images into disjoint regions, where pixels belonging to the same region are assigned the same label [10].

In simple terms, image segmentation involves dividing an image into multiple segments to filter out irrelevant information and enhance the analysis and processing of useful elements.

The significance of image segmentation extends to various fields. In the medical domain, for instance, CT images are used to assess the health of different organs in the human body. In daily life, image-based inspection systems are employed to identify targets, such as vehicles at entry and exit points. As a result, the need for efficient and accurate image segmentation has become a pressing concern.

Currently, most image segmentation methods tend to be effective only for specific problems, and general approaches are still under exploration [11]. Take the U-Net architecture as an example. U-Net is generally used in the medical field. Many situations include checking the spine [12], evaluating the function of the small intestine [13], checking the fundus and its blood vessels [14], and observing the hydrocephalus of the baby's ventricles [15]... all need to use U-Net architecture for image segmentation. In other fields, U-Net is rarely used. Over the past few decades, numerous effective methods have been developed. One of the pioneering deep learning-based image segmentation techniques employed a fully convolutional network (FCN). Additionally, the encoder-decoder architecture [16] has gained popularity for segmenting images. With the improvement of FCN and encoder-decoder architecture, better methods such as SegNet [16] and RefineNet [17]have emerged, which significantly improve the segmentation performance. However, the quest for more generalized and efficient image segmentation methods remains an ongoing area of research.

B. Unsupervised learning

Unsupervised learning is a machine learning method that automatically discovers patterns and structures in data by analyzing and processing data without relying on pre-labeled information.

In the context of image segmentation, the emergence of unsupervised segmentation was a natural progression. Initially, researchers proposed unsupervised methods for color image segmentation, although primarily used for grayscale images [11]. Subsequently, another method based on convolutional neural networks (CNNs) for unsupervised image segmentation was used the proposed CNN assigns labels to pixels that denote the cluster to which the pixel belongs [18]. Additionally, generative adversarial networks (GANs) have also been used for unsupervised image segmentation due to their properties [19].

Despite the growing activity in unsupervised segmentation, the field is still in its early stages and faces certain limitations.

Recognizing these challenges, [9]embarked on research to address these shortcomings. They explored an innovative approach known as Hypersphere-based Dimensionality Conversion (HDC). Unlike traditional ML algorithms that process data directly, HDC encodes data into a high-dimensional space, representing the data using high-dimensional and pseudo-orthogonal hypervectors (HVs). For an HV with dimension d, it can be denoted as $\vec{H}$ = ($\vec{e_1}$, $\vec{e_2}$, ..., $\vec{e_i}$, ..., $\vec{e_d}$), where $\vec{e_i}$ represents the $i-th$ element in HV $\vec{H}$. This encoding process facilitates data representation and feature extraction in the high-dimensional space. Prior studies have demonstrated the efficacy of HDC in image classification [9].

C. On device learning

On-device learning, also known as edge learning or edge intelligence, refers to machine learning applications and model training processes conducted directly on the device without relying on cloud machines. This approach aims to address the limited resources of edge devices, including computing power, memory capacity, I/O bandwidth, and energy consumption. Achieving a high-performance on-device learning system requires considerations of reducing computing overhead, improving hardware efficiency, and conserving energy costs [20]. Its key objectives encompass resource-saving, personalized models, privacy protection, and online learning.

On-device learning finds applications in various fields. In smart medical care, device-side learning facilitates functions like medical device monitoring, patient diagnosis, and drug recommendation. For intelligent transportation, on-device learning enables vehicle automatic driving, traffic signal optimization, and road condition prediction [20]. Through on-device learning, these systems achieve real-time data processing, decision-making, and feedback, enhancing the speed and accuracy of their responses [20]. Additionally, Yang et al.'s article highlights various methods and technologies related to on-device learning, including deploying traditional machine learning models like support vector machines (SVM) directly on edge devices and exploring model training techniques such as "int8" quantization for on-device learning.

However, on-device learning also faces notable drawbacks. Firstly, the limited user data on the device may not suffice for traditional distributed machine learning training. Secondly, the processing speed and memory capacity of the device might be inadequate for handling large-scale learning applications.

To improve on-device learning, several aspects require attention, such as data management and privacy protection mechanisms, flexible online learning, and incremental learning

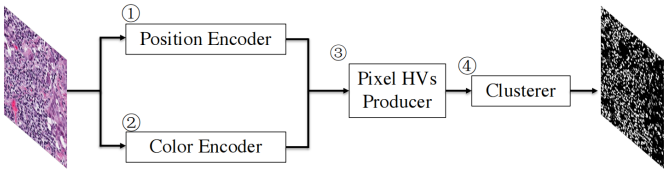

Fig. 1. Overview of the SegHDC framework [9].

technologies. Additionally, addressing the issues of insufficient data and model generalization ability is crucial [20].

These challenges create certain gaps between unsupervised segmentation and on-device learning, necessitating innovative solutions to bridge this divide. The following section will delve into the innovative content of the SegHDC framework [9], providing a specific description of its features.

III. Background

In this section, I mainly introduce the SegHDC framework, which I rely on. The framework of SegHDC [9], as depicted in Figure 1. It consists of four key sections: 1. Position Encoder Module, 2. Color Encoder Module, 3. Pixel HV Producer, and 4. Clusterer. I will briefly introduce each component in the rest part of this section.

A. Position Encoder

In any image, spatial information is the key, and naturally, its importance is self-evident in image segmentation. So how to get the spatial information of pixels from the image? Following the elaboration of the SegHDC framework [9], they use the Manhattan distance to represent the spatial relationship between pixels. Manhattan distance is a commonly used distance measurement method for measuring the distance between different positions in two-dimensional space, which refers to the sum of the absolute values of the differences between two vectors in each dimension. When p and q these two points are extended to n-dimensional space, the Manhattan distance equation remains unchanged, and the distances are superimposed sequentially, from 1 to n, where p = (p_1, p_2, ..., p_n), q = (q_1, q_2, ..., q_n).

By calculating the Manhattan distance between pixels, the location information can be encoded into a hypervector (HV) which can clearly represent and measure the relationship and difference between pixels at different locations, and provide spatial feature representation for image segmentation tasks. In addition, the authors increase the Manhattan distance by element flipping the HV, so that pixels at different locations have different representations in the HV.

Since the same flipped sites in row and column at the same time, the distances are not ideal. In order to avoid the phenomenon of decreasing distance, the rows and columns need to flip different positions. For example, flipping half of the elements in the row and flipping the other half of the elements in the column will not affect each other, and the distance will naturally become more accurate. They also introduced two other encoding methods, one called decayed Manhattan distance encoding, which describes smaller distances by introducing a new hyperparameter α to control the proportion of dimensions that need to be changed. In addition, the closer the pixels are, the more similar they are, and the more likely they are classified into one category. In order to avoid this situation they also introduced another hyperparameter β, which regards the β row and β column as a block and calculates the Manhattan distance based on these blocks, thus obtaining Block decay Manhattan distance coding.

B. Color Encoder

In the color encoder, the color values are first quantized into 256 single-channel color values, and then the Manhattan distance of the color vector is calculated using the Hamming distance and XOR operation. Adds the Manhattan distance to the color vector by flipping consecutive elements of the color HV. Finally, 256 high-dimensional vectors of 256 single-channel color values are obtained. When it extends to 3-channel images The HV encoding method can associate two HVs through element-level XOR or multiplication, but these methods cannot maintain the Hamming distance. In order to keep the Hamming distance and contain all the color information, the researchers proposed a new encoding method, which is to reduce the HV dimension of each channel from d to d_3, and then connect the HV of the three channels together to form a New HV to represent the color value of the pixel. This preserves the Hamming distance and enables efficient representation of color information.

To sum up, the function of the color encoder is to encode the color information of the image into HV for subsequent processing and analysis.

C. Pixel HV Producer

The function of the pixel HV producer is simply to maintain the Manhattan distance by performing an XOR operation on the position HV and the color HV after obtaining the position HV and the color HV, and generating the pixel HV representing the pixel. The only caveat is that color HV and position HV are pseudo-orthogonal.

$$d_c(\vec{y},\vec{z}) = 1 - \frac{\vec{y}\vec{z}}{\|\vec{y}\|\,\|\vec{z}\|} = 1 - \frac{\sum_{i=1}^{d}\vec{y}_i\vec{z}_i}{\sqrt{\sum_{i=1}^{d}\vec{y}_i^2}\,\sqrt{\sum_{i=1}^{d}\vec{z}_i^2}} \quad (1)$$

D. Clusterer

Clusterer refers to an algorithm for clustering pixel HV. The role of Clusterer is to cluster all pixels HV into different categories and calculate the center point of each category to achieve image segmentation.

The clustering algorithm used in SegHDC is K-Means, and the distance function is cosine distance (Equation 1 is as follows). The cosine distance was chosen as the distance function because the cosine distance only pays attention to the angle between the vectors, not the length of the vector, which can avoid the influence of the vector length on the distance,

and the use of the cosine distance to calculate the distance between the pixel HV and the center point HV is also to avoid the influence of the vector length on the distance.

In the clustering process, first, SegHDC selects the points with the largest pixel color differences as the initial center points, then calculates the distances between each pixel HV and the center point HV, and classify them into the closest category, and finally adds all the HVs in the same category to get the new center point HV

IV. Method

In this section, I will propose my framework ImSegHDC. The ImSegHDC has the same overview framework as the SegHDC, which is shown in Figure 1.

A. Position Encoder

As I introduced in the previous section, the position encoder of ImSegHDC undertakes the same function as in SegHDC, which aims to encode the spatial information of the pixels. The basic principle of the position encoder is to map each pixel at a different position into a high-dimensional space to make it vectorized, so as to achieve the purpose of converting position information into vector information. The specific implementation is to randomly assign a binary vector to each row and column, and use the XOR operation to associate the vectors between the rows and columns to generate a position vector.

SegHDC is proposed to use the hyperparameter β to let β rows and β columns be used as a block and calculate the Manhattan distance based on these blocks.

But in reality, the length and width of the image may be different. In this case, using one parameter to represent the column and row may produce poor results. Based on this reality, I will use two hyperparameters to control the length and width of the blocks, which may accommodate the original image size. In addition, the length and width of the picture are different, so the number of unit elements N_{col} and N_{row} are also different (shown in Equation 2).

$$x_{row} = \left\lfloor \frac{d}{N_{row}} \right\rfloor, \; x_{col} = \left\lfloor \frac{d}{N_{col}} \right\rfloor \tag{2}$$

In order to make the most use of the flip units to represent the distance, I may need to adjust the width and length at the same time. As shown in Figure 2, Figure 2 (a) shows the distances when SegHDC deal with the non-square rectangular image, which uses the square block. While Figure 2 (b) shows the relative distances when ImSegHDC deals with the non-square rectangular image, which uses the corresponding rectangular blocks. At this time, the ratio is 2 to 3, and the distances are shown in the figure. Different images can be applied with different hyperparameters to correspond to the image size.

Based on this change, the non-square images can be better accommodated with performance improvement.

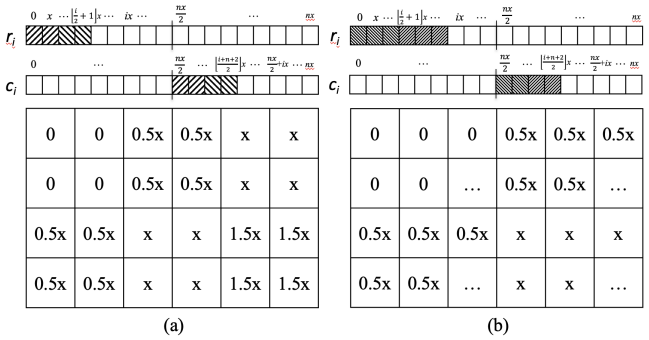

Fig. 2. Distances in SegHDC and ImSegHDC when the non-square rectangular image is used

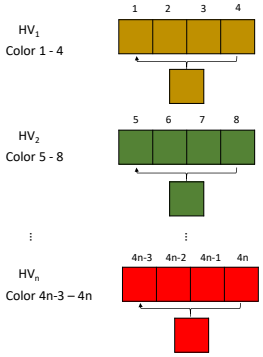

Fig. 3. Similar colors are assigned to an HV

B. Color Encoder

In addition to position information, color is also basic information for image segmentation.

In the SegHDC framework, the process of converting color information into vector information is realized through hyper-high-dimensional computing (HDC). For single-channel color values, HDC randomly generates 256 HV to represent 256 color values. If it is multi-channel, after obtaining the color value of each channel, connect the corresponding HVs together to form a new HV to represent the color value. Or randomly flips a few elements based on the previous HV. These 256 HVs are called HV_0, HV_1, HV_2, ..., HV_{255} one by one, and the distance between each pair is equal.

According to my improvement, because similar colors are more likely to be classified into one class, I set a parameter, and each parameter forms a new HV as a group (the parameter is preferably a multiple of 256 for easy calculation) so that the number of HVs will become less. In doing so, firstly, the running speed can be improved, and secondly, edge devices can use fewer vectors to achieve running, so as to achieve better results. Because according to the experiments of Yang et al., when the number of vectors is greater than 800, the effect is not good.

As shown in Figure 3. Four similar colors are assigned to one HV, and the total number of HVs changes from 4n to n.

C. Clusterer

SegHDC framework revised the K-Means to cluster the HVs which uses cosine distance to accommodate HVs. To boost the clustering convergence speed, the authors choose to use the pixels with the greatest color differences as the first batch of centroids. However, the pixels with the greatest color differences are not likely to be close to the centroids at last. If I consider the image as an even distribution, the centroids, at last, should be near the points at $\frac{1}{4}$ and $\frac{3}{4}$ of the total number of colors. Thus, in my design, if n centroids are needed, the HV is divided into n parts, and a compromise point is selected. The way of the centroids in the first iteration is chosen as Equation 3.

$$Centroid_i = \frac{i}{2n}, \quad where\ i\ is\ odd \tag{3}$$

In Equation 3, the n represents the number clusters and i is the odd number which is less than $2n$. Pixels selected based on this are more accurate and, better yet, can improve performance.

V. Experiment

A. Experimental Setup

One important point to note is the utilization of the dataset. In order to compare the difference between the method of [9] and my method, I will use the same dataset as [9], namely BBBC005 (the first two hundred images) [21], DSB2018 (stage1−train set) [22]and MoNuSeg test set [23].

It's also important to bring up the training settings. All the data in the SegHDC framework, including parameters and cluster numbers, remains unchanged. The fact that this invariance exists adds a sense of certainty to the experiment. The updated framework has made changes to all three data sets. They now have a ratio of 0.8 for the half-dimensional elements that can be flipped. Additionally, two parameters have been modified - one has been changed to 28 while the other has been changed to 22. There is a difference between the DSB2018 (stage1−train set) and MoNuSeg test sets compared to the BBBC005 (first two hundred pictures) test set. The former groups 8 colors together with a dimension value of 768, whereas the latter only groups 2 colors together and changes the dimension value to 2000. The remaining data is identical to the SegHDC framework.

The baseline method used in this study is identical to the one employed by Yang et al. The evaluation method utilizes Intersection over Union (IoU) metrics. IoU is the ratio of the intersecting area between the forecasted segmentation map and the actual ground truth, divided by the combined area of both elements [9].

B. Experimental Results

Figure 4 shows the sample image, ground truth, prediction mask of baseline, SegHDC, ImSegHDC, and the IoU score obtained by the three methods. For example the BBBC005 sample in the first row, the IoU score of the ImSegHDC framework is slightly higher than that of the SegHDC framework.

TABLE I
IoU score on 3 datasets

Dataset	BL [18]	SegHDC	ImSegHDC	Improvement
BBBC005	0.7490	0.9414	0.9330	24.6% ↑
DSB2018	0.6281	0.8038	0.7958	26.7% ↑
MoNuSeg	0.5088	0.5509	0.5326	4.7% ↑

TABLE II
Result of latency on Raspberry Pi for processing an image in DSB2018 dataset and BBBC dataset

	Image Size	Latency on PI	SpeedUp
Baseline	256 × 320 × 3	11453.0s	baseline
SegHDC	(DSB2018)	35.8s	319.9×
ImSegHDC		**17.22s**	**665.1** ×
Baseline	520 × 696 × 1	× *	baseline
SegHDC	(BBBC005)	178.31s	-
ImSegHDC		**69.47s**	-

× * Out of memory.

For the DSB2018 samples in the second row,The IoU score of the ImSegHDC framework is nearly 0.6 higher than that of the SegHDC framework. While in the MoNuSeg samples in the third row, the IoU score of the ImSegHDC framework is slightly lower than that of the SegHDC framework. But the IoU scores of both frameworks are higher than the baseline in three samples.

Table I displays the IoU scores for the three datasets. I referenced the data from the baseline and SegHDC methods tested by [9], and compared them to my own method, ImSegHDC.

The analysis of these findings contributes to drawing significant conclusions. The ImSegHDC method outperforms the baseline, as evidenced by its significantly higher IoU score.

Furthermore, while the IoU score for ImSegHDC may be lower than SegHDC on the surface, careful calculation reveals that ImSegHDC's reduction ratio in the MoNuSeg dataset is below 4%, and less than 1% in the two datasets of BBBC005 and DSB2018. Although the average IoU score of ImSegHDC is slightly lower than that of SegHDC, its effectiveness remains noteworthy.

While ImSegHDC may have a slightly lower IoU score on average compared to SegHDC, this does not necessarily mean that all images will have a high IoU score. In some pictures, the performance of ImSegHDC is better than that of SegHDC, as shown in Figure 4. Figure 4 shows sample images, ground truth, baseline, SegHDC, ImSegHDC, and the IoU values of these three methods. The IoU score of ImSegHDC is higher than SegHDC in BBBC005 and DSB2018 datasets, and lower than SegHDC in MoNuSeg dataset.

Table II shows the latency results obtained by processing images on the Raspberry. According to the dataset for DSB2018, the SegHDC has a latency of 35.80 seconds, while the ImSegHDC has a latency of 17.22 seconds. This means that the latency of ImSegHDC is 2.1 times higher than that of SegHDC. In the BBBC005 dataset, the latency of

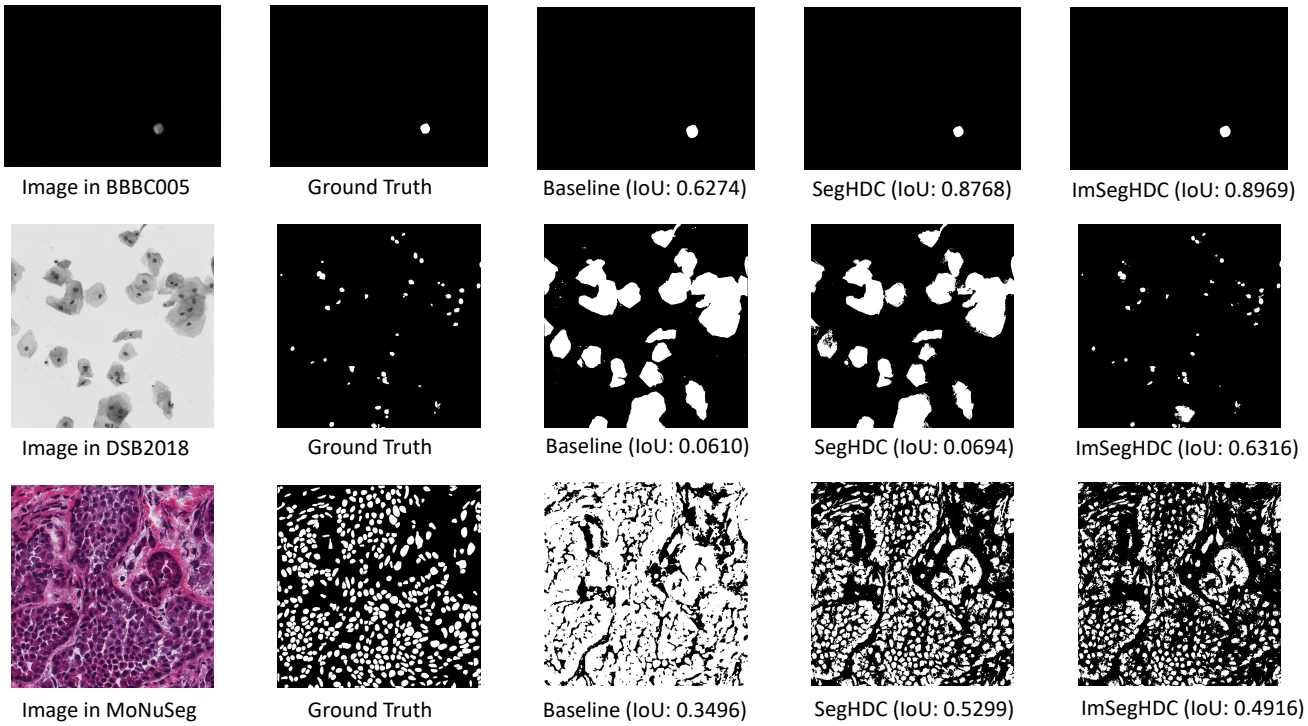

Fig. 4. Images from BBBC005, DSB2018 and MoNuSeg datasets

SegHDC is 178.31 seconds, while that of ImSegHDC is 69.47 seconds, which is 2.6 times higher than that of SegHDC. In addition, it needs to be briefly mentioned that the dimension of ImSegHDC is a little lower than that of SegHDC.

It becomes clear that while the IoU score of ImSegHDC is slightly lower than SegHDC, ImSegHDC has significantly less delay. Overall, the ImSegHDC framework outperforms SegHDC.

VI. Conclusion

On the basis of the SegHDC framework, it has been improved to produce the ImSegHDC framework. After conducting the IoU score test and delay test on the identical data set, it was determined that the ImSegHDC framework slightly outperformed the SegHDC framework when both judges' results were combined. The ImSegHDC framework improves unsupervised image segmentation on devices.

References

[1] S. Dabeer, M. M. Khan, and S. Islam, "Cancer diagnosis in histopathological image: Cnn based approach," *Informatics in Medicine Unlocked*, vol. 16, p. 100231, 2019.

[2] A. I. A, M. N. A, U. M. A, T. M. A, M. S. H. B, I. P. C, A. R. D, and A. I. A, "Towards using cough for respiratory disease diagnosis by leveraging artificial intelligence: A survey," 2022.

[3] P. Dixit, P. Bhattacharya, S. Tanwar, and R. Gupta, "Anomaly detection in autonomous electric vehicles using ai techniques: A comprehensive survey," *Expert Systems*, 2021.

[4] O. K. Tonguz and R. Zhang, "Methods and systems for self-organized traffic management at intersections using a distributed ai approach," 2020.

[5] G. Barbeito, D. Budde, M. Moll, S. Pickl, and B. Thiebes, *A Hybrid AI and Simulation-Based Optimization DSS for Post-Disaster Logistics*. Advances in Security, Networks, and Internet of Things, 2021.

[6] Famili and A., "Use of decision-tree induction for process optimization and knowledge refinement of an industrial process," *Artificial Intelligence for Engineering Design Analysis Manufacturing*, vol. 8, no. 01, pp. 63–75, 1994.

[7] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham: Springer International Publishing, 2015, pp. 234–241.

[8] F. Shi, J. Wang, J. Shi, Z. Wu, Q. Wang, Z. Tang, K. He, Y. Shi, and D. Shen, "Review of artificial intelligence techniques in imaging data acquisition, segmentation, and diagnosis for covid-19," *IEEE Reviews in Biomedical Engineering*, vol. 14, pp. 4–15, 2021.

[9] J. Yang, Y. Sheng, Y. Zhang, W. Jiang, and L. Yang, "On-device unsupervised image segmentation," 2023.

[10] E. Anjna and E. R. Kaur, "Review of image segmentation technique," in *National Conference on Emerging Trends on Engineering Technology (ETET-2017)*, 2017.

[11] Z. Cao, X. Zhang, and X. Mei, "Unsupervised segmentation for color image based on graph theory," in *2008 Second International Symposium on Intelligent Information Technology Application*, 2009.

[12] N. Shigeta, M. Kamata, and M. Kikuchi, "Effectiveness of pseudo 3d feature learning for spinal segmentation by cnn with u-net architecture," *Journal of Image and Graphics*, vol. 7, no. 3, pp. 107–111, 2019.

[13] K. Otsuki, Y. Iwamoto, Y.-W. Chen, A. Furukawa, and S. Kanasaki, "Cine-mr image segmentation for assessment of small bowel motility function using 3d u-net," *Journal of Image and Graphics*, vol. 7, no. 4, pp. 134–139, 2019.

[14] A. O. Joshua, F. V. Nelwamondo, and G. Mabuza-Hocquet, "Blood vessel segmentation from fundus images using modified u-net convolutional neural network," *Journal of Image and Graphics*, vol. 8, no. 1, pp. 21–25, 2020.

[15] K. Ono, Y. Iwamoto, Y.-W. Chen, and M. Nonaka, "Automatic segmentation of infant brain ventricles with hydrocephalus in mri based on 2.5

d u-net and transfer learning," *Journal of Image and Graphics*, vol. 8, no. 2, pp. 42–46, 2020.

[16] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[17] G. Lin, A. Milan, C. Shen, and I. Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation," 2016.

[18] W. Kim, A. Kanezaki, and M. Tanaka, "Unsupervised learning of image segmentation based on differentiable feature clustering," *IEEE Transactions on Image Processing*, vol. 29, pp. 8055–8068, 2020.

[19] R. Abdal, P. Zhu, N. J. Mitra, and P. Wonka, "Labels4free: Unsupervised segmentation using stylegan," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 13 950–13 959.

[20] Q. Zhou, Z. Qu, S. Guo, B. Luo, and R. Akerkar, "On-device learning systems for edge intelligence: A software and hardware synergy perspective," *IEEE Internet of Things Journal*, vol. PP, no. 99, pp. 1–1, 2021.

[21] Sokolnicki, L. K., Carpenter, E. A., and Ljosa, "Annotated high-throughput microscopy image sets for validation."

[22] E. P. j.-n. J. K. M. N. P. S. W. C. Allen Goodman, Anne Carpenter, "2018 data science bowl," 2018. [Online]. Available: https://kaggle.com/competitions/data-science-bowl-2018

[23] N. Kumar and et. al, "A multi-organ nucleus segmentation challenge," *IEEE Transactions on Medical Imaging*, vol. 39, no. 5, pp. 1380–1391, 2020.