Quality Evaluation of Image Segmentation in Mobile Augmented Reality

Shneka Muthu Kumara Swamy(✉) and Qi Han

Colorado School of Mines, Colorado, USA
{smuthukumaraswamy,qhan}@mines.edu

Abstract. Mobile Augmented Reality (AR) facilitates a seamless interactive experience between actual and virtual environments. AR employs segmented images for various purposes such as object recognition, occlusion boundary estimation, and foreground-background separation. However, evaluating the quality of segmented images in mobile AR is challenging due to the limited resources of mobile devices. Existing solutions employ neural networks with many layers, making it difficult to deploy them on mobile devices. To address this issue, we propose techniques to modify the inputs so that we can reduce the number of layers in the neural network, making it possible to deploy for mobile devices. This idea is incorporated into our proposed SegQNet. It utilizes deep learning techniques based on convolutional neural networks (CNNs) to evaluate the quality of overlaid segmentation in mobile AR. SegQNet achieves high accuracy without the need for ground-truth images or expensive computations. Our experiments on Android smartphones demonstrate that SegQNet outperforms two state-of-the-art methods without incurring significant overhead.

Keywords: Augmented Reality · Quality of Segmentation · Neural Network for Mobile Devices

1 Introduction

Mobile Augmented Reality (AR) has gained immense popularity in recent years, offering the ability to overlay digital images onto real-world environments. Image segmentation plays a vital role in AR applications, providing a more immersive and realistic user experience. In particular, AR uses segmented images for object recognition, occlusion boundary estimation, and foreground-background separation [14,16,17]. For example, when a user is navigating through a crowded urban environment with multiple routes to choose from. The user can benefit from an overlaid map and a segmented view of different real-world objects such as streets, buildings, and sidewalks.

Despite the significant progress made in enhancing image segmentation techniques for many applications, the evaluation of segmentation quality remains

A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 594, pp. 415–427, 2024.
https://doi.org/10.1007/978-3-031-63992-0_27

a challenging issue for mobile augmented reality (AR). The currently available methods for assessing the quality of segmentation are not entirely suitable for mobile AR since they either rely on ground truth images that may not always be accessible or require computationally intensive operations that mobile devices may not support. Therefore, a fast and resource-efficient quality evaluation method suitable for mobile AR is needed.

Evaluating the quality of segmented images in mobile AR is faced with several challenges. First, mobile devices have limited computational resources, and any quality assessment process should not compete with the primary AR application for resources. Consequently, evaluation methods must be designed to be efficient in their use of resources. Second, the storage capacity for each Android app is restricted, making it impractical to store a large deep-learning model. Therefore, a lightweight model needs to be developed to complete the task. Three, AR applications may require quantitative quality measurement, so a rough quality classification may be inadequate. Last, AR scenes are complex as they may contain occlusion, background, and foreground objects.

This paper tackles these challenges by presenting a fast and efficient method for evaluating segmentation quality suitable for mobile AR. To demonstrate that our method can work for mobile AR, we implement it on Android smartphones. More specifically, our main contributions in this work are as follows.

1. We have designed SegQNet, a no-reference model that utilizes a lightweight convolutional neural network (CNN) to determine the quality of segmented images. SegQNet distinguishes itself from existing work in several ways. 1) SegQNet does not require many layers in the neural network, making it possible to deploy on mobile devices. 2) Unlike some previous studies [12,22] that were limited to medical images, SegQNet can handle all kinds of images. 3) SegQnet has the ability to process images of varying sizes, eliminating the need to pre-process images to conform to a specific size as required in existing methods [19]. 4) SegQNet does not need ground truth images.
2. We have evaluated SegQNet's performance on a PC with a GPU and compared it to two other approaches – MultiScale Net [19] and Pull the Plug [12]. The results show that SegQNet achieves an average loss that is comparable to MultiScale Net but uses less storage and computation. SegQNet also outperforms Pull the Plug in terms of average loss.
3. We have implemented SegQNet, MultiScale Net, and Pull the Plug on Android smartphones. Results show that SegQNet is more efficient in terms of CPU and energy usage and incurs less delay compared to MultiScale Net.

The rest of the paper is organized as follows. Section 2 provides a summary of the related work; Sect. 3 discusses our model for evaluating the quality of segmented images overlaid in AR; In Sect. 4 we present the performance of SegQNet and comparison with two baseline approaches using GPU in a PC; In Sect. 5 we demonstrate the working of our model in smartphones and show it performs better than baseline approaches. We conclude our paper in Sect. 6.

2 Related Work

Image segmentation is susceptible to six types of errors [8], including dilation, erosion, boundary localization, object localization, boundary approximation, and inner mask.

Objective segmentation quality evaluation methods can be classified into full reference (i.e., requiring the comparison against ground truth images), no reference (i.e., no need for ground truth images), and partial reference methods (i.e., no need for ground truth images, but need some kind of base images). The most recent full reference methods [18] considered human visual perception rather than solely comparing pixels.

Several no-reference approaches utilize machine learning methods to predict segmentation quality. Some methods use segmentation properties such as image edges, boundary, and coverage to evaluate quality [12], but this approach was designed for medical images that have their own unique characteristics. In contrast our approach works for general images. Other methods use only the original and segmented image and implement neural networks for determining the quality. Examples include neural networks [13,19], transfer learning [13], a three cascaded layer of double nets with five convolutional layers [19]. However, these methods are not developed for resource-constrained mobile devices, resulting in high storage and execution time, which is not suitable for smartphones. In addition to the segmented and the original image, some other no-reference neural network methods [10] assume the presence of spatial uncertainty maps. Yet other neural network methods such as [6] predict a newly defined quality metric. Instead, we adopt the commonly used evaluation metric Intersection over Union (IOU) and do not assume the presence of spatial uncertainty maps.

Segmentation can be done for single or multiple regions. Thus, quality can also be determined for single or multiple regions. For example, [5] considers the quality of only the region of interest to the viewers. Our work considers the quality of multiple-region segmentation. Unlike single region segmentation, which only considers the proper segmentation of a single object, multiple region segmentation considers multiple objects as well as the background.

No-reference segmentation quality evaluation that uses supervised learning algorithms can further be categorized as classification and regression methods. Classification methods [11,13] determine the quality as 'Good', 'Ok', and 'Bad'. They determine the quality by setting a threshold for each class in the method (e.g.,'Good' is from 0.8 to 1.0, 'Ok' is from 0.4 to 0.8). These thresholds vary from user to user and are specific to applications. In contrast, regression methods [12, 19] generally return a value from 0 to 1, with 0 being the worst quality. Thus, we consider the quality of segmentation as a regression problem in this paper.

3 Neural Network-Based Evaluation of Image Segmentation Quality

In this section, we present our design of Segmentation Quality Network (SegQNet) that evaluates the quality of segmented images overlaid in AR scenes.

SegQNet adopts CNN [15], a commonly used neural network for letter identification, object detection, and the optical flow of images. A CNN has an input layer, a hidden layer followed by fully connected layers, and an output layer. The input layer consists of images to be processed. The hidden layer performs convolution operations on these images using a kernel of fixed size. Fully connected layers are then used for invariance and fine-tuning of the network. The output layer provides the final result as a value in our quality evaluation.

SegQNet (Fig. 1) consists of two convolutional layers and two fully connected layers. We added an adaptive average pool layer to make sure that the network can handle images of different sizes. In addition, we increase the kernel size of the network from 5×5 to 10×10 for performing a reduced number of operations. We next present the details of the input and output layers in SegQNet.

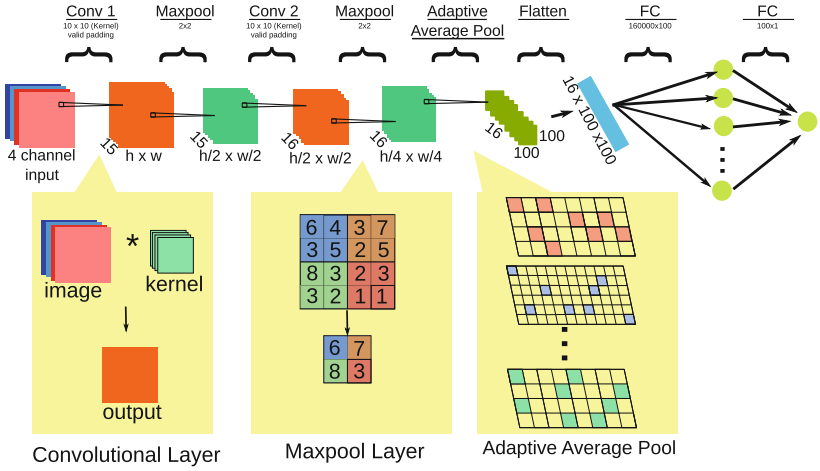


Fig. 1. Neural Network Architecture of SegQNet. The top part shows the overall architecture and the bottom part shows a snippet of the working of each layer.

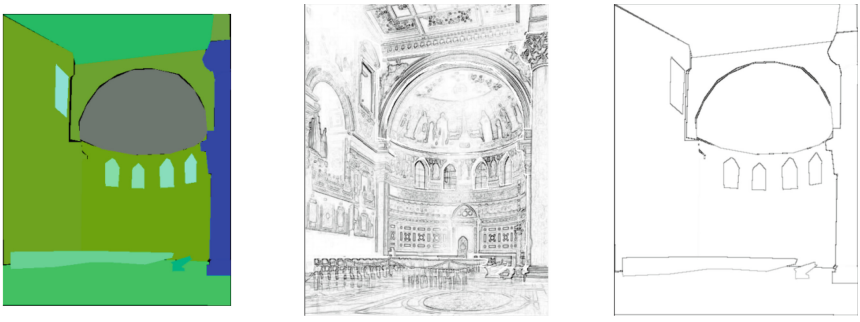
Input Layer. Since the segmentation of an image chooses which edges to keep, we use the edge images of both the original and the segmented image to determine the inputs to SegQNet. There are various methods to determine the edge of a given image, such as Sobel, Canny, and Laplacian filters. A Sobel filter works under the principle that the presence of an edge can be determined by the color changes in the adjacent pixels. The color changes are obtained by the gradients of an image (a.k.a. edge intensity), which is the difference between the adjacent pixel values. We use the edge intensity to determine whether an edge is present or not. We use a Sobel Filter instead of Canny or Laplacian filters. This is because Canny or Laplacian filters operate on the output of the Sobel filter and choose the most probable edges where misclassification may occur.

The Sobel filter is applied to both the original and segmented images to obtain their edge intensity and orientation. The edge orientation, which refers

to the direction of the gradient, is also used as an input to SegQNet. We expect the orientation of the segmented edge and the original edge to be nearly identical with proper segmentation.

The difference between the edge intensities of the original and segmented image and vice versa are calculated. Figures 2 and 3 show the difference between the original and the segmented image and vice versa. The difference in edge intensity between the original and segmented images could be zero, positive, or negative. Zero difference suggests the presence of edges in both the original and segmented images. A positive difference indicates an edge exists in the original image but not in the segmented image. A negative difference indicates the presence of an edge in the segmented image but not in the original image. Since this negative difference becomes positive when we compute the difference between the edge intensities of the segmented and the original images, we discard negative differences to simplify the representation.

Figures 2b and 3b show examples of white pixels, which indicate the absence of edges or edges in the segmented image that were not present in the original image. In Fig. 3b, a white line indicates an extra edge in the Dilated image, that should actually not be present (Inside the block shown in red). By comparing Figs. 2b and 2c, we can see that the intersection of all white pixels represents places where there are no edges, while the difference represents the extra edges. We use these two inputs with the orientation and feed them to the CNN which has the ability to utilize spatial correlation to learn the quality.



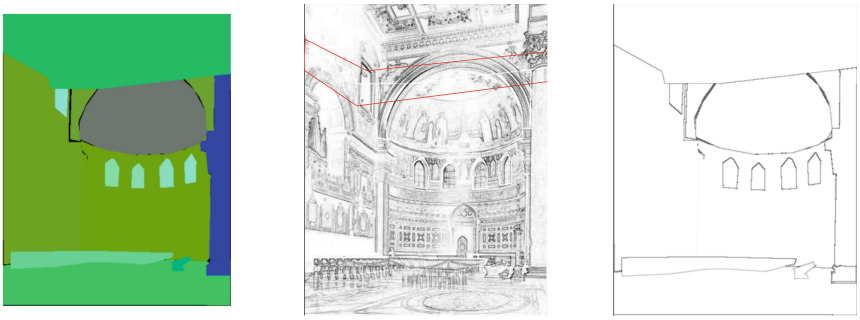
(a) Original image segmented

(b) original_sobel - segmented_sobel

(c) segemented_sobel - original_sobel

Fig. 2. Input to SegQNet when the ground truth image segmented is given as input where original_sobel is the original image edge intensity and segmented_sobel is the segmented image edge intensity ((b) and (c) are inverted for better visualization)

Based on earlier studies [23], it has been observed that neural networks primarily learn low-level characteristics such as edges and colors in the initial layers. Given that our input already utilizes edge images, we can employ a neural network with fewer layers. Moreover, our proposed methodology of finding the difference between edge intensity of the original and segmented images enables



(a) Dilated image segmented

(b) original_sobel - dilated_sobel

(c) dilated_sobel - original_sobel

Fig. 3. Input to SegQNet when the dilated image segmented is given as the input, where original_sobel is the original image edge intensity and segmented_sobel is the segmented image edge intensity ((b) and (c) are inverted for better visualization)

SegQNet to identify the existence of extra edges, which is an essential aspect in evaluating image quality. Hence, we constructed SegQNet utilizing a simpler architecture like LeNet.

In summary, we provide the original image edge orientation, the segmented image edge orientation, the original image edge intensity subtracted from the segmented edge image intensity, and the segmented image edge intensity subtracted from the original image edge intensity as inputs to SegQNet.

Output Layer. This layer provides a single value as the image quality. Traditional metrics for segmentation quality are DICE (Jaccard Score) [9], precision and recall, Mask Intersection over Union (IoU) [21], and Boundary IOU [8]. More recent research has designed a new metric that considers human visual perception. However, all these methods are full-reference method (i.e., need ground truth images). Therefore, we decided to use IOU, the most commonly used metric [2], which ratio of the number of pixels in the segmented image with errors introduced that match the ground truth image over the total number of pixels in both images.

4 Performance Evaluation on a PC with GPU

We first evaluate the performance of SegQNet on a PC with GPU and compare that with baseline approaches. We utilize the ADE20k dataset [1,24,25] in our evaluation. This dataset includes ground truth segmented images as well as original images. We selected the ADE20k dataset also because it contains multi-object segmentation as well as both foreground and background segmentation. Furthermore, this dataset offers extensive coverage of both indoor and outdoor images, which is where AR is predominantly used. The dataset contains about 434,826 object instances, 2,693 object classes, and 9.9 object classes per image. ADE20k dataset has 20,210 images in the training set, 2,000 images in the val-

idation set, and 3,000 images in the test set. We use the same training-test set partition as specified in the dataset for our evaluation.

When considering the right approach to compare against, we do not consider the neural network algorithms [11,13] that treat quality analysis as a classification problem. This is because we treat quality evaluation as a regression problem and we would like to provide a quantitative value for the quality. Methods [6] that define a new quality metric are not fair to be compared against, either. Therefore, we choose two most relevant algorithms as our baselines: (1) MultiScale Net [19] that uses a neural network model, and (2) Pull the Plug [12] that uses a traditional machine learning model.

- **MultiScale Net.** Two neural network-based models [19] was developed for the quality evaluation of segmented images. The first is a double-network model that has two cascaded VGG16 nets with the original and segmented images as inputs. The second is a MultiScale Net with six or eight cascaded VGG16 layers with original and segmented images in the actual and also cropped sizes as inputs. The results demonstrate the superiority of the MultiScale Net over the double-net model, hence we implement the MultiScale Net for our comparison. However, Multiscale Net uses VGG architecture that can only accept images of one size (i.e., 224×224). Images of 224×224 and the corresponding segmented images are fed to the first two cascaded layers. Also, according to the original Multiscale Net paper, for the other four layers, the original and the segmented image must be cropped first and then interpolated back to 224×224 for VGG. We utilize the same details as described in the paper in our implementation for fairness of comparison. Since their paper does not mention the size of the smaller images, we consider the common cropping sizes, 112×112 and 168×168. It is important to note that, we use the same images as input to our SegQNet, but do not make modifications to the size of the image as our network can work with images of different sizes.
- **Pull the Plug.** Pull the Plug [12] was developed for a single object in the foreground. In contrast, our work considers multiple object segmentation. For a fair comparison, we randomly choose one object out of all the objects in the images for quality evaluation.
  In addition, Pull the Plug considers a bounding box and convex hull around the segmented object. While this is valid for most of the error types considered, it is not always true for erosion. As applying erosion to a small object can make the object disappear completely from the scene. The diminished object creates a bounding box and convex hull with a minimal area close to zero. Therefore, we do not include these sets of images in our experiments of Pull the Plug.

For comparison, we generate negative samples by modifying the segmented images to include the six common types of errors described previously. We trained and tested all three algorithms (i.e., our SegQNet, MultiScale Net, and Pull the Plug) using NVIDIA 3090Ti GPU and 20-core Intel I7 processor.

We trained SegQNet on the different types of errors in segmentation. To provide a reasonable comparison, we trained MultiScale neural network and Pull

the Plug as mentioned in their papers. Table 1 compares the average error of the three different quality evaluation methods: SegQNet, MultiScale Net, and Pull the Plug.

Our findings demonstrate that SegQNet's accuracy in evaluating image quality is equivalent to MultiScale Net, but surpasses Pull the Plug by a significant margin. However, it is critical to consider the complexity of the neural network models utilized, specifically MultiScale Net and SegQNet. SegQNet achieves comparable accuracy to MultiScale Net for an input image size of 224×224 (a commonly used size in machine learning algorithms), but with much lower memory requirements (30 times less), a smaller number of trainable parameters (10 times less), and substantially fewer GFLOPs (45 times less) as shown in Table 2. We attribute the better performance of SegQNet mainly to the inherent differences in the architecture. Multiscale Net uses many layers and is likely to determine the edge of the images after initial layers [23] and hence requires more layers. On the other hand, in our approach, edge detection is done first, hence SegQNet requires fewer layers, resulting in smaller GFLOPS, and less memory requirement.

Table 1. Average error of quality evaluation

Method	Average Error
SegQNet	0.07
MultiScale Net	0.07
Pull the Plug	0.41

Table 2. Complexity of the Neural Network models of size 224×224

Method	Total Memory(MB)	Total trainable params	No. of mult-add (GFLOP)
MultiScale Net	2164.64	214,967,169	92.626
SegQNet	71.70	16,030,232	2.11

Further, as shown in Table 3, when processing images with resolutions similar to those used by ARCore (640×480) [3], SegQNet has lower complexity compared to MultiScale Net.

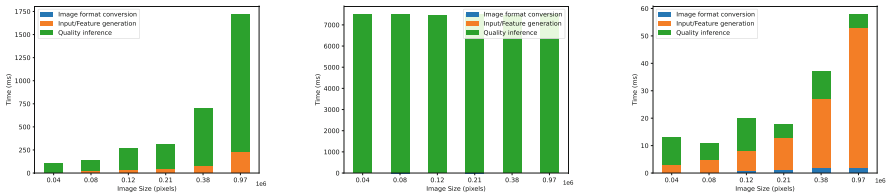
Table 3. Complexity of the Neural Network models of size 640×480

Method	Total Memory(MB)	Total trainable params	No. of mult-add (GFLOP)
MultiScale Net	2164.64	214,967,169	92.626
SegQNet	113.55	16,030,232	5.05

5 Performance Evaluation on a Smartphone

Since the main aim of this work is to enable quality evaluation for mobile AR, we implemented SegQNet, MultiScale Net, and Pull the Plug on OnePlus 6T smartphones. We used DeepLabv3 [7] provided by PyTorch for image segmentation

first before running each of the quality evaluation methods. To ensure that the neural network loaded onto phones is lightweight, we utilize various optimization techniques such as quantization, pruning, operator fusion, kernel optimization, and model compression on both SegQNet and MultiScale Net. These methods are implemented in PyTorch Lite and do not compromise the accuracy of the neural network. These optimizations are crucial for enabling the models to run efficiently on mobile devices without sacrificing their performance.



(a) SegQNet (b) MultiScale Net (c) Pull the Plug

Fig. 4. Latency breakdown comparison

Latency. Fig. 5a and Fig. 5b highlight that the time taken to execute the quality evaluation utilizing SegQNet is roughly 83.3 times shorter than the segmentation process itself. This result is significant since it is undesirable for quality evaluation to consume more time than the segmentation algorithm itself.

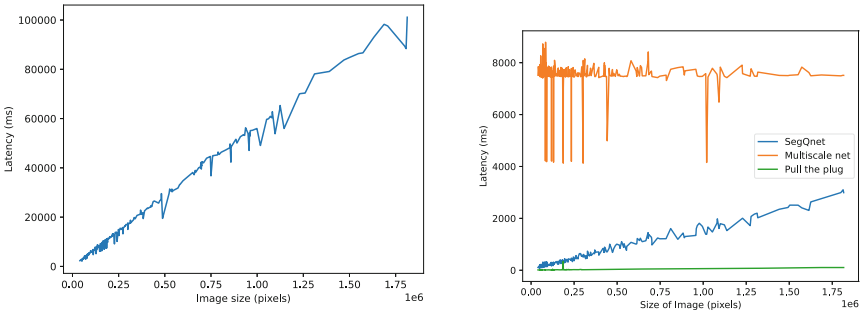
Figure 5b also shows that the time taken by SegQNet is about seven times less than MultiScale Net, so SegQNet can achieve comparable performance as MultiScale Net but with much less time. In fact, MultiScale Net takes more time than the segmentation itself for images of smaller sizes. However, SegQNet takes more time than Pull the Plug, especially for larger images. This can be attributed to the number of convolution operations performed in SegQNet. However, the additional few milliseconds of the extra time taken by SegQNet significantly improves its accuracy, i.e., by six times when compared with Pull the Plug as shown in Table 1. By examining Fig. 5b, it is evident that SegQNet's quality determination process takes up to 3 s for a few large images. However, these images are typically high-resolution and not frequently used. In fact, the MultiScale Net architecture decreases all images to 224×224 using VGG16. By restricting the image size to this small dimension, SegQNet requires less than 250 milliseconds to assess the quality. Additionally, when we consider the resolution employed by ARCore to process video for AR applications, SegQNet's execution time is less than 300 milliseconds.

Figure 4a shows the breakdown of the time for quality evaluation. It takes very little time to convert Bitmap to 'Mat'. This processing time is incurred because of the need to convert from Bitmap, the type for reading an image in Java, to 'Mat', the required type for OpenCV, and back to Bitmap, the type required for PyTorch. Most of the processing time in SegQNet is dedicated to the neural network and the computation of edge intensity and orientation of

images. Due to the neural network's significant role and the absence of image scaling, the SegQNet processing time grows as the input image size increases, as the number of convolution operations also increases.

Figure 4b depicts the breakdown of the time to run MultiScale Net. This network has the same conversion latency as SegQNet. The 'Input/feature generation' is the time taken to rescale the images to the size required by the network. Most of the time taken by this approach is to run the network.

Figure 4c is the breakdown for time to run Pull the Plug. In this case, more amount of time is taken to get the features than to run the trained model. This is because to get the features, the algorithm needs to traverse through the image, whereas for deriving inference, it runs through the linear regression model. It is also intuitive that the latency to calculate the features increases with the size of the image.



(a) (b)

Fig. 5. (a) Image segmentation Latency (b) Quality evaluation Latency

Energy consumption. Battery usage for performing segmentation for 1500 images uses about 95% of the smartphone battery, this involves rendering the image to the screen as well. According to the paper [20], the maximum amount of energy/battery utilization in an AR app is due to camera sampling and rendering. The battery utilized to render the image to the screen is not the consideration of the paper, and hence we measure the battery utilization after removing the rendering. Table 4 compares the battery utilization of three different quality evaluation methods for 1,000 images. SegQNet battery utilization is comparable to Pull the Plug, and it is about 12 times less than that of MultiScale Net.

CPU Usage. We recorded the total CPU usage for 1,000 images and observed that SegQNet used 50% of CPU for a short period of time in the beginning and then stayed at about 30%. The similarity of CPU usage is because SegQNet and MultiScale Net do not vary in the complexity of each operation performed but vary only in the number of instances the operations are performed. However, there is a clear difference in the total latency involved. The time duration that the CPU runs at 30% for SegQNet is similar to Pull the Plug but significantly less than MultiScale Net.

Table 4. Battery Utilisation of quality evaluation

Method	Battery Utilization
SegQNet	4 %
MultiScale Net	48 %
Pull the Plug	1 %

Memory Usage. The models trained by the neural network are optimized for phones and then stored in phones. After the training, the memory required by SegQNet is about 60 MB, which is about 14 times less than MultiScale Net. We applied a quantization process to reduce the model size of MultiScale Net, but it still takes about 500 MB, which is quite demanding for a smartphone. We use ONNX [4] to store the Linear Regression model of Pull the Plug, and it takes only 600 B. Therefore, Pull the Plug takes the least space among the three, but it is also at the cost of the lowest accuracy in quality evaluation.

6 Conclusion

In this paper, we presented our techniques for determining the quality of overlaid segmented images in mobile AR apps. We determine the quality based on the most common six types of errors that may appear in the segmentation: dilation, erosion, object localization, boundary approximation, boundary localization, and inner mask errors. We developed SegQNet, a lightweight CNN that has an average error of 0.07. Our experiments show that SegQNet achieves approximately six times less absolute error when compared to Pull the Plug. While SegQNet shows comparable accuracy with MultiScale Net, SegQNet incurs much less overhead. Experiments on smartphones demonstrate that overall SegQNet outperforms existing approaches when considering performance and cost trade-offs. SegQNet cannot determine under- or over-segmentation. This is acceptable considering that recent advances in segmentation techniques [8] have eliminated under- or over-segmentation as a type of error. With SegQNet being fast and resource-efficient, We are in the process of developing a framework that integrates real-time quality evaluation with adaptive segmentation in mobile AR.

References

1. Ade20k dataset. https://groups.csail.mit.edu/vision/datasets/ADE20K/
2. Papers with code - semantic segmentation. https://paperswithcode.com/task/semantic-segmentation

3. Recording and playback introduction, arcore, google developers. https://developers.google.com/ar/develop/recording-and-playback
4. Sklearn-onnx: Convert your scikit-learn model into onnx, https://onnx.ai/sklearn-onnx/
5. Cai, Z., Liang, Y., Huang, H.: Unsupervised segmentation evaluation: an edge-based method. Multimedia Tools Appl. **76**(8), 11097–11110 (2017)
6. Chen, H., Peng, B., Al-Huda, Z., Li, X.: Metric learning with feature embedding for segmentation quality evaluation. In: 2021 16th International Conference on Intelligent Systems and Knowledge Engineering (ISKE), pp. 269–275. IEEE (2021)
7. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587 (2017)
8. Cheng, B., Girshick, R., Dollár, P., Berg, A.C., Kirillov, A.: Boundary IOU: Improving object-centric image segmentation evaluation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 15334–15342 (2021)
9. Crum, W.R., Camara, O., Hill, D.L.: Generalized overlap measures for evaluation and validation in medical image analysis. IEEE Trans. Med. Imaging **25**(11), 1451–1461 (2006)
10. DeVries, T., Taylor, G.W.: Leveraging uncertainty estimates for predicting segmentation quality. arXiv preprint arXiv:1807.00502 (2018)
11. Feng, T., Peng, B., Al-Huda, Z., Jia, H.: Segmentation quality evaluation algorithm based on classification. In: 2021 16th International Conference on Intelligent Systems and Knowledge Engineering (ISKE), pp. 254–261 (2021). https://doi.org/10.1109/ISKE54062.2021.9755428
12. Gurari, D., Jain, S., Betke, M., Grauman, K.: Pull the plug? predicting if computers or humans should segment images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 382–391 (2016)
13. Huang, C., Wu, Q., Meng, F.: Qualitynet: segmentation quality evaluation with deep convolutional networks. In: 2016 Visual Communications and Image Processing (VCIP), pp. 1–4. IEEE (2016)
14. Kim, W., Seok, J.: Indoor semantic segmentation for robot navigating on mobile. In: 2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN), pp. 22–25. IEEE (2018)
15. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
16. Li, X., Tian, Y., Zhang, F., Quan, S., Xu, Y.: Object detection in the context of mobile augmented reality. In: 2020 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp. 156–163. IEEE (2020)
17. Perry, J., Fernandez, A.: MineNet: a dilated CNN for semantic segmentation of eye features. In: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (2019)
18. Shi, R., Ma, J., Ngan, K.N., Xiong, J., Qiao, T.: Objective object segmentation visual quality evaluation: quality measure and pooling method. ACM Trans. Multimedia Comput. Commun. Appl. (TOMM) **18**(3), 1–19 (2022)
19. Shi, W., Meng, F., Wu, Q.: Segmentation quality evaluation based on multi-scale convolutional neural networks. In: 2017 IEEE Visual Communications and Image Processing (VCIP), pp. 1–4. IEEE (2017)
20. Wang, H., Xie, J.: User preference based energy-aware mobile AR system with edge computing. In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications, pp. 1379–1388. IEEE (2020)

21. Wang, Z., Wang, E., Zhu, Y.: Image segmentation evaluation: a survey of methods. Artif. Intell. Rev. **53**(8), 5637–5674 (2020)
22. Zaman, F.A., Zhang, L., Zhang, H., Sonka, M., Wu, X.: Segmentation quality assessment by automated detection of erroneous surface regions in medical images. In: Computers in Biology and Medicine (2023)
23. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8689, pp. 818–833. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10590-1_53
24. Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Scene parsing through ade20k dataset. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 633–641 (2017)
25. Zhou, B., et al.: Semantic understanding of scenes through the ade20k dataset. Int. J. Comput. Vision **127**(3), 302–321 (2019)