PortraitNet: Real-time Portrait Segmentation Network for Mobile Device

Leonardo Pesce 223100001
CIE6004: Image Processing and Computer Vision

***Abstract*—The paper "*PortraitNet: Real-time Portrait Segmentation Network for Mobile Device*" [1] introduces PortraitNet, a real-time portrait segmentation model designed for mobile devices. It utilizes a lightweight U-shape architecture with two auxiliary losses during training, specifically the boundary loss and consistency constraint loss, to improve accuracy and robustness. PortraitNet outperforms existing methods in accuracy and efficiency, particularly in generating sharper boundaries and handling challenging lighting conditions. Additionally, it can process 224 × 224 RGB images at 30 FPS on an iPhone 7 in real-time.**

I. Introduction

The paper discusses the growing importance of real-time portrait segmentation, driven by the demand for mobile applications that enable background editing of portrait images. It highlights the challenges posed by the unique characteristics of portrait images, such as the need for precise segmentation in the presence of ambiguous boundaries and varying illumination conditions. Existing methods have prioritized accuracy over efficiency, making them unsuitable for real-time mobile applications.

To address these challenges, the authors propose PortraitNet, a specialized convolutional neural network designed for real-time portrait segmentation on mobile devices. PortraitNet utilizes a specific network architecture with a 32× downsampling rate to enhance efficiency and employs a U-shape architecture for better segmentation. The network is trained using auxiliary losses, including a boundary loss to improve boundary accuracy and a consistency constraint loss to enhance robustness under varying illumination.

The results demonstrate that PortraitNet achieves high accuracy, with 96.62% accuracy on the EG1800 dataset and 93.43% on the SupervisePortrait dataset. Furthermore, it can process images at 30 frames per second on an iPhone 7 with an input size of 224 × 224, making it suitable for real-time mobile portrait segmentation.

II. Method

A. Model Structure

Let's now have a look at the model implemented in the paper. The model is based on the MobileNet-v2 [2], which is an architecture specifically designed for mobile devices (low computational power). The MobileNetV2 design employs an inverted residual structure, which differs from traditional residual models by utilizing slender bottleneck layers for both input and output within the residual block. Instead of expanded representations in the input, MobileNetV2 employs lightweight depthwise convolutions to filter features in the intermediate expansion layer.

The structure is U-shape with a x32 downsampling in the encoder phase that combined with the residual architecture extracts global and spatial information about the image.

B. Auxiliary Losses

To make up for the lower complexity of the model (since it needs to run in real-time) two auxiliary losses are introduced.

1) Boundary Loss: Given that, portrait segmentation needs sharper edges and that the edge represents less than 10% of the labeled dataset we use the focal loss to balance this fact. Focal loss L_e is combined with cross-entropy loss L_m in the following way:

$$L = L_m + \lambda L_e \tag{1}$$

$$L_m = -\sum_{i=1}^{n}(y_i log(p_i) + (1-y_i) log(1-p_i)) \tag{2}$$

$$L_e = -\sum_{i=1}^{n}(1-p_i)^{\gamma} y_i log(p_i)) + p_i^{\gamma}(1-y_i) log(1-p_i)) \tag{3}$$

Where γ is the weight of the boundary loss, and the probability p_i is computed as

$$p_i(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{4}$$

2) Consistency Constraint Loss: In general, in a segmentation process, hard labels (binary labels) are used. Given our complexity requirements, since it has been shown that for small models soft labels help in the training process, a new method to generate soft labels is proposed (usually generating soft labels requires a teacher model that needs extensive training). Given an input A a second input A' is generated simply by a texture enhancement transformation (change in brightness, contrast, sharpness, noise...). At this point, both images are passed through the model obtaining the output B and B'. B' is worse than B because it was generated by A', a lower quality version of A. In the end, B is used as a higher quality soft label for B' and the KL-divergence is used to compute the loss between the two:

$$L = L'_M + \alpha \times L_c \tag{5}$$

$$L_c = \frac{1}{n}\sum_{i=1}^{n} q_i \times log\frac{q_i}{q_i'} \times T^2 \tag{6}$$

$$\begin{aligned} L_m' = &-\sum_{i=1}^{n}(y_i log(p_i) + (1-y_i)log(1-p_i)) \\ &-\sum_{i=1}^{n}(y_i log(p_i') + (1-y_i)log(1-p_i')) \end{aligned} \tag{7}$$

Where α is used to balance the two losses, T to smooth the output, p_i and p_i' are:

$$p_i(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \qquad p_i'(z_j') = \frac{e^{z_j'}}{\sum_{k=1}^{K} e^{z_k'}} \tag{8}$$

and q_i and q_i' are:

$$p_i(z_j) = \frac{e^{\frac{z_j}{T}}}{\sum_{k=1}^{K} e^{\frac{z_j'}{T}}} \qquad p_i'(z_j') = \frac{e^{\frac{z_j'}{T}}}{\sum_{k=1}^{K} e^{\frac{z_k'}{T}}} \tag{9}$$

Adding this loss results in higher accuracy and robustness under different lighting conditions.

C. Pseudo-code

We will now analyze the pseudo-code of the model and training: The model training follows the typical structure. The dataset was split into 3 with train, test, and final evaluation. The first two are used during the epochs of the model training phase to compute new parameters of the network and check if the values are better than before. The last is used to compute the final accuracy of the model.

Algorithm 1 pseudocode for model training

```
computes the learning rate
executes a training cycle
executes a testing phase to update the loss
save the model if the loss is the lowest between all the cycles
```

The structure of a training cycle is as follows.

Algorithm 2 pseudocode for the training cycle

```
for every element in train set do
    i=image, ie=image enhanced m=mask, and e=edge
    mi, ei = network(i) (mask i, edge i)
    mie, eie = network(ie) (mask ie, edge ie)
    loss += softmax_loss(mi, m)
    loss += focal_loss(ei, e) * edge_ratio
    loss += softmax_loss(mie, m)
    loss += focal_loss(eie, e) * edge_ratio
    loss += KL_loss(mi, mie)
    loss += KL_loss(ei, eie)
    computes the gradient and updates the parameters
end for
```

The first loss is the one between the mask computed on the image and the real mask, the second is the loss computed between the predicted edges on the image and the real edges. The next two are the same but the prediction was made with the enhanced image. The last two are the Kullback-Leibler Divergence loss between the mask and edge predicted on the real image and the mask and edge predicted on the enhanced image.

The test cycle is simply composed by the sum of all the Intersection Over Union (IoU) computed on the test set.

Algorithm 3 pseudocode for the testing cycle

```
for every element in test set do
    i=image and m=mask
    mi, ei = network(i) (mask i, edge i)
    total_IoU += IoU(mi, m)
end for
return total_IoU
```

III. Results

The model trained for 1000 epochs with a batch size of 64, a learning rate of 0.001, momentum of 0.9, and weight decay of 0.0005. The final model accuracy (tested on 130 images) was 94.97%, the accuracy metric was IoU on the real and computed masks. In the following graph, we can see the training loss curve, which follows a typical inverse function decreasing sharply at the beginning and reducing the convergence speed gradually.

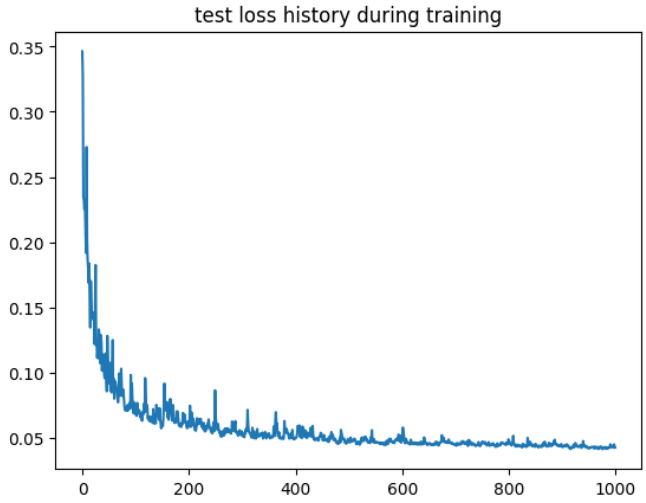

Fig. 1: example 1

We now run our model on the dataset provided EG1800 and test its accuracy and results. For every example there are 4 images, the first and the third contain the predicted mask and edge of the person, while the second and the fourth contain the real mask and edge.

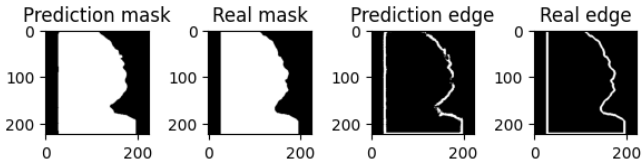

Fig. 2: example 1

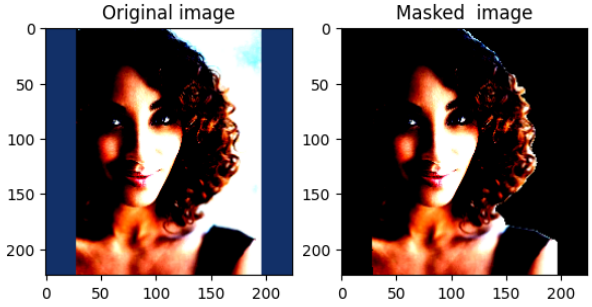

Fig. 3: mask result of example 1

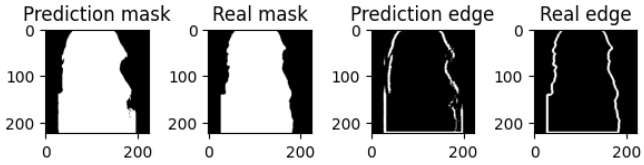

Fig. 4: example 2

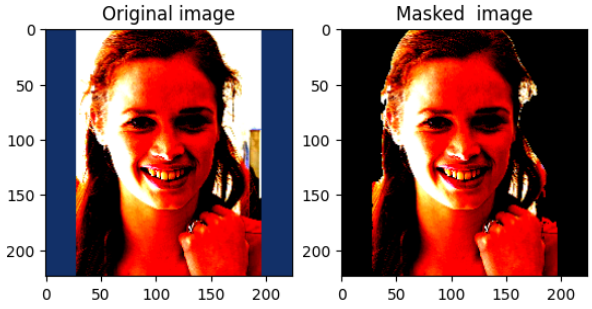

Fig. 5: mask result of example 2

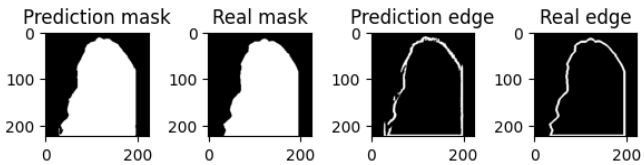

Fig. 6: example 3

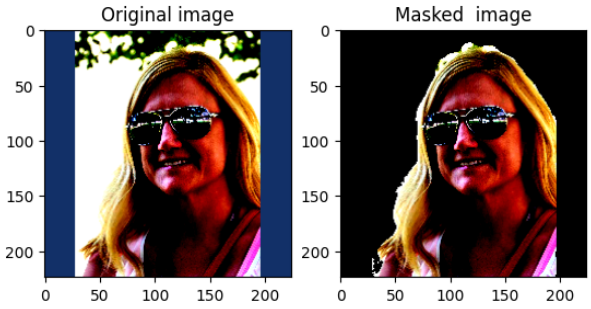

Fig. 7: mask result of example 3

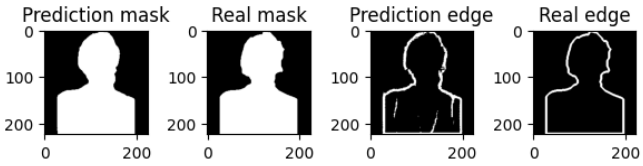

Fig. 8: example 4

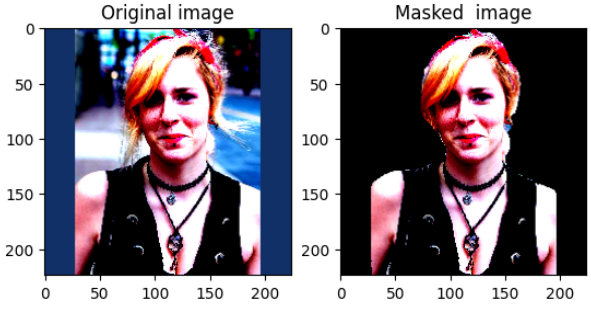

Fig. 9: mask result of example 4

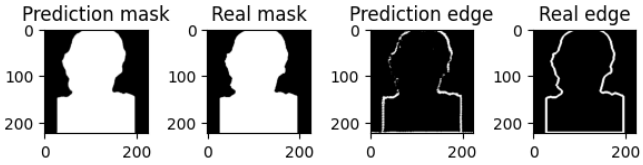

Fig. 10: example 5

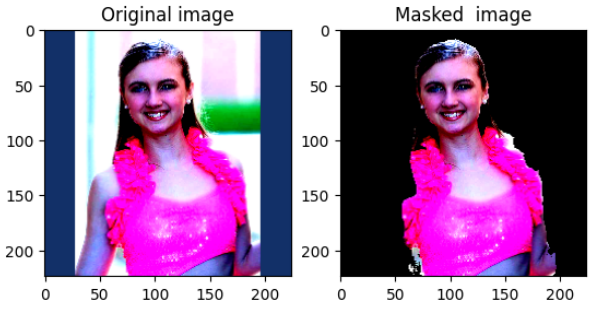

Fig. 11: mask result of example 5

A. Encountered problems

The method proposed required a long training (at least a long training for my PC without a GPU). The problem was solved by optimizing as much as possible the code and using Google Colab in order to use the free GPU resources available. Also instead of training for 2000 epochs we only trained for 1000 epochs (which given the curve **??** shouldn't change the results).

IV. Conclusions

The paper discusses the creation of a light and fast neural network architecture based on MobileNet-v2 to segment people inside the picture (with a target on mobile devices). Of fundamental contribution in this paper besides the structure of the network are also the boundary loss and the consistency constraint loss. We analyzed the structure and mathematical understanding of the paper and finally proposed an implementation of it to validate the results proposed by the author.

References

[1] S.-H. Zhang, X. Dong, H. Li, R. Li, and Y.-L. Yang, "Portraitnet: Real-time portrait segmentation network for mobile device," *Comput. Graph.*, vol. 80, pp. 104–113, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:131924823

[2] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation," *CoRR*, vol. abs/1801.04381, 2018. [Online]. Available: http://arxiv.org/abs/1801.04381