# Image Processing and Computer Vision (CSC6051/MDS6004) Assignment 1

Zhang Shiqi
224040081
224040081@link.cuhk.edu.cn

## Abstract

*The results of each task will be displayed below.*

1. *Task1*

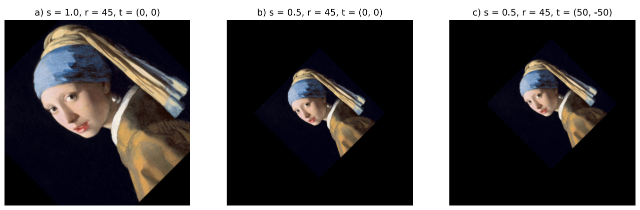
a) s = 1.0, r = 45, t = (0, 0)
b) s = 0.5, r = 45, t = (0, 0)
c) s = 0.5, r = 45, t = (50, -50)

Figure 1. Combined affine result

2. *Task2*

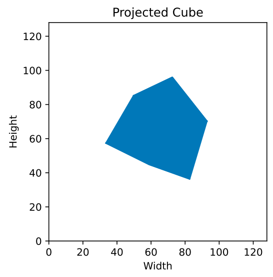

Figure 2. Default settings of Task2 plot

3. *Task3*

Table 1. Experiments result by PortraitNet

| Dataset | Method | Test on | Metric |
|---|---|---|---|
| EG1800 | | EG1800 | 0.9608 |
| EG1800 | 2000 Epoch | EG1800 | 0.9653 |
| EG1800 | Face Detection | EG1800 | 0.9615 |
| Mattinghuman | | Mattinghuman | 0.9537 |
| Mattinghuman | | EG1800 | 0.9281 |
| EasyPortrait | eyes | EasyPortrait | 0.7790 |
| EasyPortrait | tooth | EasyPortrait | 0.7303 |

Please read the **Read_me.ipynb** file before starting the training code. From an assignment perspective, the model's training-related information is placed in the **checkpoints folder**. From a model training/testing perspective, place the model's checkpoint file in the **myexp folder**. Checkpoints was updoaded to OneDrive (**link.txt**).

## 1. Task 1

**Implementation Details** For this task, we first calculate the basic properties of the image, with the image size set to $w$ , $h$, the coordinates of the centre of the image as $centre = (\frac{w}{2}, \frac{h}{2})$, and the given angle $\theta$ and radian conversion as $radian = (\pi * \theta)/180$. Subsequently, we randomly select three localised points in the image, named $triangle\_old$, and use the formula $triangle = triangle\_old - centre$ to obtain the decentred triangle coordinates.

Next we solve the affine transformation matrix. First, we use the formula

$$rotation\_matrix = \begin{pmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{pmatrix}$$

to compute the rotation matrix of the triangle, and then we use the formula

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

to complete the triangle rotation and translation process. Next, we use the $cv2.getAffineTransform()$ function to compute the old and new triangle affine transform matrices $AT\_matrix$. Finally, the function $cv2.warpAffine()$ is used to get the final changed image.

**Results & Analysis** We reproduced the example given in the title in our experiments, and from Fig.1 a) we can observe that when scale and translation distance are constant,

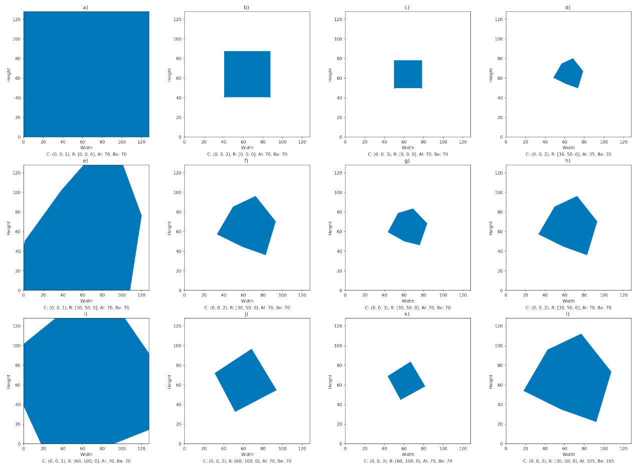

Figure 3. The result of different setting models

$\theta > 0$, the picture rotates counterclockwise, and $\theta < 0$, the picture rotates clockwise. From Fig. Fig.1 b), we can also observe that when $scale < 1$, the image size will be smaller than the original size, and when $scale > 1$, the image size will be larger than the original size. From figure Fig.1 c) we can observe that the image pans to the right when $t_x > 0$, down when $t_y > 0$, and vice versa when $t_x < 0, t_y < 0$. (Note that at this point the x-axis and y-axis positive directions are right and down, The figure was showed in Abstract part.)

## 2. Task 2

### Implementation Details

1. Get camera intrinsics matrix $K$: In order to build camera intrinsics matrix $K$, we use this function:

$$K = \begin{bmatrix} \alpha & 0 & c_x \\ 0 & \beta & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

2. Returns the point in screen space $x_s$: Base on the theory of projective transformation in the homogeneous coordinate system, we can use this function to calculate $x_s$.

$$x_s = \begin{bmatrix} \alpha\frac{x}{z} + c_x \\ \beta\frac{y}{z} + c_y \end{bmatrix}$$

3. Plot the projected cube. After used for loop, we can projects the 3D point $x_c$ to screen space and returns the 2D pixel coordinates, it shape is (6, 4, 2).

   The default settings of the projected cube shows in Fig. 2.

**Results & Analysis** The default settings as illustrated in topic was showed in Fig. 2.

Fig. 3 illustrates the projection images of the same cube on a 2D plane under different conditions.

Fig. 3 (a, e, i), (b, f, j), and (c, g, k) display the changes in the cube's appearance for three different rotation angles under three sets of central positions and focal lengths. As the rotation angle changes, the orientation and perceived depth of the cube shift significantly, at larger angles, more of the cube's edges and faces become visible, creating a stronger sense of 3D structure in the 2D projection.

Fig. 3 (a-c), (e-g), and (i-k) illustrate the changes in the cube's appearance for three different central positions under three sets of rotation angles and focal lengths. At off-center positions, the cube appears more distorted, and certain faces dominate the view depending on the shift direction. When the cube is centered, its appearance maintains balance, with fewer visible distortions in its geometric structure.

Fig. 3 (d, h, l) show the changes in the cube's appearance for three different focal lengths under three sets of rotation angles and central positions. With longer focal lengths, the cube appears smaller, and perspective distortion is minimized, resulting in an orthographic-like view.

## 3. Task 3

PortraitNet is a real-time portrait segmentation model for mobile devices [7]. The model uses a lightweight U-shaped structure and introduces two auxiliary losses in the training phase: boundary loss and consistency constraint loss. The model was trained and evaluated on the EG1800 and Mattinghuman datasets, respectively, to test the performance of the model under small and large datasets [6, 3]. Furthermore, the face recognition module was added to the original PortraitNet model to enhance the model's performance on facing non-central portrait segmentation. In addition, the model was trained on the EasyPortrait dataset, which in turn explored the potential of the model to segment human facial features [2].

### 3.1. The Encoder-Decoder structure of PortraitNet

In fact, the Encoder for PortraitNet model is improved from MobileNet-v2 [5]. In portrait segmentation tasks, we are dealing with people whose subjects often make up a large part of the image. In order to fully learn the global information in the image, the Encoder of the PortraitNet model consists of several Inverted Residual Blocks, which are the core modules of MobileNet-v2. The Inverted Residual Block module consists of a series of convolutions, which contains three sub-modules, which are The Inverted Residual Block consists of a series of convolutions with three submodules, namely Pointwise Convolution, Depthwise Convolution, and Pointwise-Linear Convolution, which downscales and upscales the data through a wide–narrow–wide bottleneck structure. Inverted Residual Block is characterised by low number of parameters and high computational speed. With the input image size optimised to 224x224, the Encoder of the PortraitNet model adopts a downsampling rate of 32 times to learn the global features of the image.

The Decoder module of the PortraitNet model, on the other hand, is simpler, and the main body employs several Upsample modules to upsample the Encoder's output with a total sampling rate of 32 times. Prior to the operation of each upsampling module, the PortraitNet model designs a residual network to perform residual computation with the output of each E-stage module in the Encoder module.

After the whole Encoder-Decoder network operation, we will get two outputs, one will be used to calculate the Mask loss and one will be used to calculate the Boundary loss.

The plot of the network can be found in [7] P3 Fig.2. The code of the network can be found in file PortraitNet.py line 132-188.

## 3.2. The loss functions and evaluation metrics

As mentioned above, PortraitNet uses two different loss functions, Boundary loss and Consistency constraint loss. Also, the evaluation metrics used in this paper are mean Interaction-over-Union (IOU)

**Boundary loss:** The purpose of introducing Boundary loss in PortraitNet is to allow the network to better recognise the clear boundaries of the characters in order to render better results in downstream tasks such as background replacement or background blurring. In PortraitNet, the boundary of the image (224x224) is set to a size of 4 pixels.Since more than 90% of pixels in the boundary ground truth images are negative, the representation of boundary is difficult to learn. Thus the model uses focal loss to guide the learning of boundary loss. Focal loss L is:

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

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

$$L = L_m + \lambda \times L_e \tag{3}$$

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

In the above, $L_m$ is the cross-entropy loss and $L_e$ is the focal loss. $\lambda$ is the weight of boundary loss. $y_i$ represents the ground truth label of pixel $i$. $p_i$ represents the predicted probability of pixel $i$. The predicted probability $p_i$ computed as $p_i(z_j)$, where $z$ is the original output of PortraitNet, and $K$ is the number of groundtruth classes.

**Consistency constraint loss:**The consistency constraint loss can further improve the accuracy of the model and enhance the performance of the model in terms of lupulin in different image backgrounds. Typically, the labels that are often used for character segmentation tasks are 0,1 labels, i.e., the portrait portion is labelled with 1 and the rest with 0. However, it has been shown in many articles that soft labels are often more informative, and they can be better used to help in the training of the model[1]. In PortriatNet, the authors utilised a series of image preprocessing methods to perform deformation enhancement and texture enhancement on the images. Image A is deformed and enhanced to get image A', image A' is texture enhanced to get image A". Inputting image A' and A" into PortriatNet respectively will result in two output heat maps B' and B". At this point it is easy to see that the heat map B" is less effective than B". Therefore, the authors use heatmap B' as a soft label for heatmap B". Specifically, the authors add a consistency constraint loss between heatmaps B' and B", who is denoted as KL scatter, which is expressed by the formula:

$$L'_m = -\sum_{i=1}^{n}(y_i \log(p_i) + (1-y_i)\log(1-p_i)) \tag{5}$$

$$-\sum_{i=1}^{n}(y_i \log(p'_i) + (1-y_i)\log(1-p'_i)) \tag{6}$$

$$L_c = \frac{1}{n}\sum ni = 1 q_i \times log\frac{q_i}{q'_i} \times T^2 \tag{7}$$

$$L = L'_m + \alpha \times L_c \tag{8}$$

$$p_i\left(z_j\right) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad p'_i\left(z'_j\right) = \frac{e^{z'_j}}{\sum_{k=1}^{K} e^{z'_k}} \tag{9}$$

$$q_i\left(z_j\right) = \frac{e^{\frac{z_j}{T}}}{\sum_{k=1}^{K} e^{\frac{z_j}{T}}} \quad q'_i\left(z'_j\right) = \frac{e^{\frac{z'_j}{T}'}}{\sum_{k=1}^{K} e^{\frac{z'_j}{T}'}} \tag{10}$$

where $\alpha$ is used to balance the two losses and $T$ is used to smooth the outputs.

The plot of two losses can be found in [7] P5 Fig.4.

**mean Interaction-over-Union (IOU):**The quantitative metric used to evaluate segmentation precision is the mean Interaction-over-Union(IOU) as follows:

$$mean\ IOU = \frac{1}{N} \times \sum Ni = 1 \frac{maskPD_i \cap maskGT_i}{maskPD_i \cup maskGT_i} \tag{11}$$

where $maskPD_i$ and $maskGT_i$ represent segmentation result and ground truth label of $i$–th image of test dataset, respectively.

## 3.3. Train and evaluate the model

**Data loading and preprocessing pipelines:** First, we load the corresponding images and labels of the dataset through cv2 and perform special processing on the labels in order to make the dataset conform to the portrait segmentation task. Subsequently, we perform data enhancement on the images with the following pipeline for data enhancement:random translation - rotation - scaling - rotation - color - blur - noise. After the data enhancement is completed, we resize the target image with its labels to match the 224x224 size required by the model and extract the corresponding edge detection map.

In this task, we additionally tested the performance of the model under different training batches. With a training set of EG1800 and a training batch of 200 Epoch, the model's mIOU score is 0.9608. with a training batch of 2000 Epoch, the model's mIOU score is 0.9653. The performance metrics of the model are shown in Table 2. The training loss of the model is shown in Fig. 4.

Table 2. EG1800 result by PortraitNet

| Dataset | Epoch | Test on | Metric |
|---|---|---|---|
| EG1800 | 200 | EG1800 | 0.9608 |
| EG1800 | 2000 | EG1800 | 0.9653 |

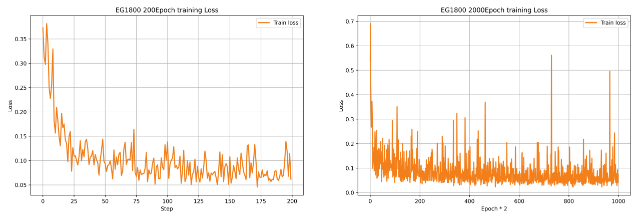

Figure 4. EG1800 training Loss comparison on different epoch

## 3.4. Demonstrate model performance on custom cases

Fig. 5 illustrates the actual effect of the network dataset on the EG1800 model. We can clearly find that when the model is at 200 Epochs, the edge processing at the right shoulder of the character in this image is not clear enough, and it seems to recognize the black stripe on the character's collar as the background, which appears to be the phenomenon of edges diverging inward. And when the model is trained to 2000 Epochs, the edge clarity is greatly improved, and the model can better deal with the problem of high contrast of the color of the picture details mentioned before. However, it is worth noting that there is a large lens flare effect on the left shoulder of the character in this picture, although the model's handling of the flare at 2000 Epochs is already better than that of the 200 Epoch model, we can still observe that while the edge of the character's left shoulder was sharpened, the model also sharpened the edge of the flare together, and a rectangular area with greenish color appeared.

## 3.5. Face Detection and Image Preprocessing

In order to enhance the performance of the model for edge cutting in non-dramatic character images, we try to introduce a face centering data preprocessing module. Specifically, we add the face centering module between the image label introduction and processing module and the image enhancement module. We used

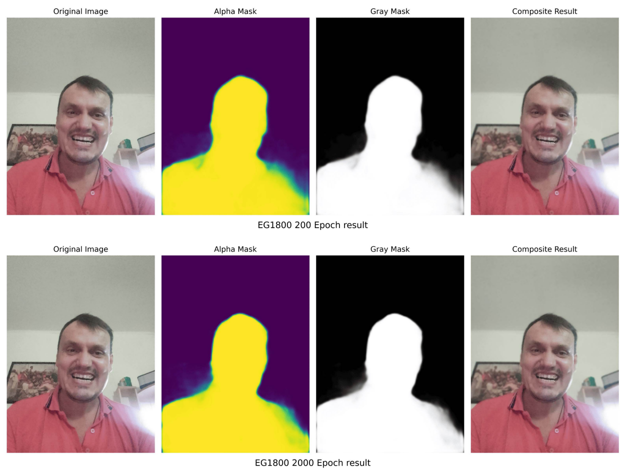

Figure 5. EG1800 model performance on custom cases

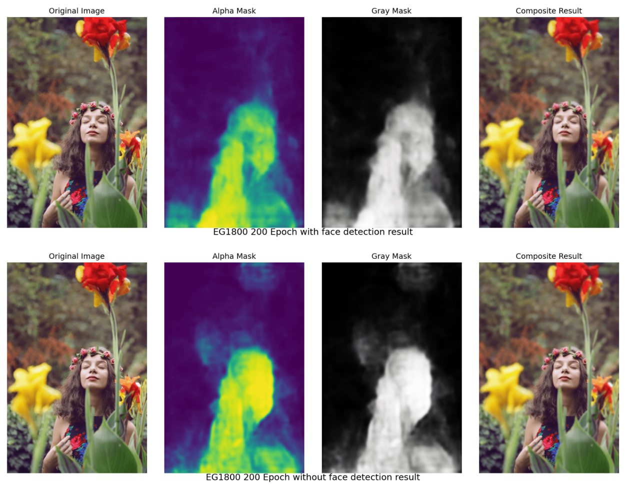

Figure 6. EG1800 model (Face Detection) performance

the $cv2.CascadeClassifier()$ function and adopted the $haarcascade\_frontalface\_default.xml$ face recognition model in the pipeline [4]. When the face recognition module recognizes a face, it first acquires the coordinates of the face and expands the region of Offset size on the coordinates in order to make the new region contain as much of the person as possible and less of the background. The expanded coordinates are then applied to the labeled image for cropping. The newly acquired portrait region will be set as the new input image.

As shown in Fig. 6, when the same image of an uncentered person is applied to two different models, it exhibits a completely different effect. We can clearly see that the base model that does not use the face centering module focuses more on the global information of the image, and it seems to identify the flowers that are similar in color to what the person is wearing with the flowers behind the person as edges. The model that used the face centering module did not have this type of problem problem. The model with the face centering module added seems to be able to focus itself more on the target than on the cluttered background when confronted with small or marginal characters. The relevant experimental results are shown in Table 3.

Table 3. EG1800 result by PortraitNet with face detection

| Dataset | Method | Test on | Metric |
|---|---|---|---|
| EG1800 | With Face | EG1800 | 0.9615 |
| EG1800 | Without Face | EG1800 | 0.9608 |

## 3.6. Large-scale Dataset Pretraining

The results of the related experiments are shown in Table 4 and the results of the model visualization are shown in Fig.7. Contrary to intuition, the performance of the model trained on large datasets seems to be inferior to the performance on small datasets. We also chose EG1800 as the benchmark test set, and as can be seen from Table X, the model trained on the EG1800 dataset has an mIOU of 0.9608, while the model trained by Mattinghuman has an mIOU of only 0.9281. Analyzing the visualization results, we can see that the model trained by Mattinghuman seems to show some hallucinatory phenomenon, which sees the picture frame in the background of the figure with the picture inside as a part of the figure, which may be related to the quality of the dataset. However, comparing the other parts of the character, Mattinghuman has better performance than EG1800 in the character's ear, the right side of the character's glare part.

When PortraitNet is trained on a larger dataset, it does bring more advantages, such as reducing the risk of model overfitting. And, more samples help to help the model learn a more comprehensive set of features, which increases the robustness of the model.

When operating on large datasets, the model will often require more parameters for capturing the features of the data. At the same time, the model will also require more computing resources, such as GPU resources. All the experiments in this paper are done on a 2080Ti. It took about 22 hours to train 2000 Epochs with the dataset EG1800, while it took almost 26 hours to train 200 Epochs on the Mattinghuman dataset.

Table 4. Experiments result by PortraitNet on Mattinghuman

| Dataset | Epoch | Test on | Metric |
|---|---|---|---|
| EG1800 | 200 | EG1800 | 0.9608 |
| Mattinghuman | 200 | Mattinghuman | 0.9537 |
| Mattinghuman | 200 | EG1800 | 0.9281 |

## 3.7. Face Parsing Extension

The performance of the model trained under the EasyPortrait dataset is shown in Table 5. Fig. 8 illustrates the

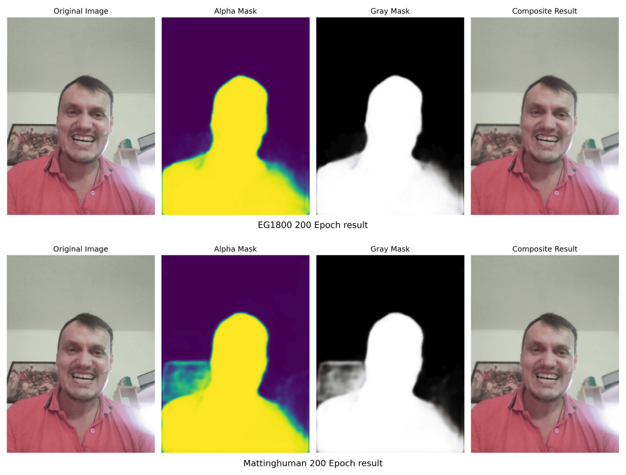

Figure 7. Mattinghuman model performance on custom cases

Table 5. Face features experiments result by PortraitNet

| Dataset | Facial features | Test on | Metric |
|---|---|---|---|
| EasyPortrait | eyes | EasyPortrait | 0.7790 |
| EasyPortrait | tooth | EasyPortrait | 0.7303 |

results of the model on the realistic dataset. The performance of the model trained under the EasyPortrait dataset is shown in Table X. Figure X illustrates the results of the model on the realistic dataset. The model visualizes better during the eye segmentation task, but sometimes recognizes the mouth as an eye as well. During the tooth segmentation task, the model visualization is not so clear with no visible edges.

Next, a brief description of the changes that need to be made to the output layer and loss function when the model is to be trained on the EasyPortrait training set for face parts. When we are training the teeth model, the shape of the output layer and loss function do not need to be changed because there are still only two classes in the dataset (teeth and background). But when we need to train on eyes, we need to change the n_class in the code to 3 because the labels in the dataset are left eye + right eye, which are different labels. In other words, the output layer shape needs to be changed from nn.Conv2d(self.depth(8), 2, 3, 1, 1, 1, bias=False) two classes to nn.Conv2d(self.depth(8), 3, 3, 1, 1, 1, bias=False) three classes.

In the loss function part, when the output value is larger than two categories, Focal Loss needs to first adjust the input data to the relevant shape. We define the significance of the following variables: N–Batch, C–Channel, H–Height, W–Width. when the input feature channel¿2, we first need to change the data from N,C,H,W to N,C,H*W. and then change the data from N,C,H*W to N,H*W,C. Finally, change the data from N,H*W,C to N*H* W,C. In this way, Focal Loss can support more classes of loss calculations.

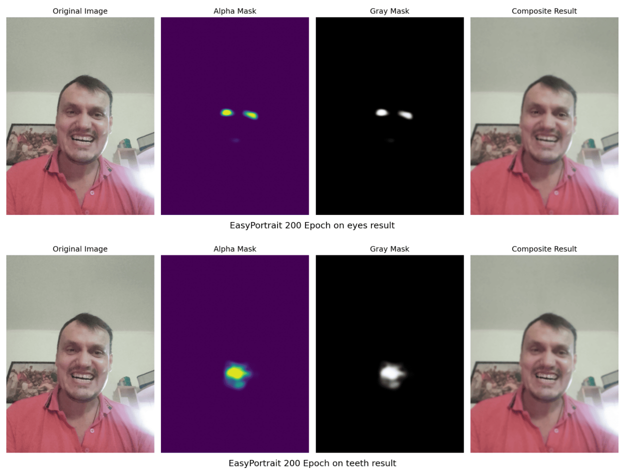

Figure 8. Experiments result by PortraitNet on EasyPortrait

## 4. Conclusion

In this assignment, we completed Image Affine Transformation and the results are shown in Fig. 1. We also completed the Project 3D Points to Retina Plane assignment and the results are shown in Fig. 2 and 3. The assignment that took the most time was Task 3: Replicating PortraitNet and Improving the Network. The results of the PortraitNet for different methods and different datasets are shown in Table 1. Fig. 5,6,7,7,8 show the visualization of the different models respectively. We described the Encoder-Decoder structure of the PortraitNet network in our homework and explained the loss function of the model in detail. In our experiments we found that:

1. The larger the training Epoch, the better the model works.

2. PortraitNet improved based on the face recognition module can be better optimized for the recognition of backgrounds that are similar to the person.

3. The larger the dataset, the better the visualization of the model is, but there is a possibility that the model can be hallucinatory.

4. The model has a lot of potential for the facial feature recognition task, perhaps with a Face Recognition Module improved PortraitNet network can have better results on facial feature recognition task.

## 5. Code Usage

**Start training**: 1. Set the path to the yaml file in the train.py file (lines 436–475). 2.Change data_root: (the upper level of the root directory of the target dataset), file_root: (the root directory of the .txt file that stores the label/path of the dataset), model_root(the path that stores the experimental diary and the optimal model) in the corresponding .yaml file. 3.Run the train.py file to start the training.

**Wish to start training on your own dataset**: 1. Use Build_dataset.py file to build the corresponding .txt path index file of the dataset (different datasets need to adjust the code structure a little bit). 2. Create a new .yaml file, fill in the above addresses into the data_root, file_root, model_root, change the datasetlist for the name of their own dataset. 3. in dataset.py file in the myDataSet class in the new class of data reading methods. 4. in dataset.py file in the PortraitSeg class to add new pictures and labels in the logic of the read-in.

**To test the complete test set and get the mIOU score**: 1. Set the config_path variable in the test.py file (lines 63–93) 2. Run the test.py file.

**Visualize the model single image effect**: 1. change the config_path variable in test_for_single.ipynb. 2. set the address of the image to be tested in the img_ori variable. 3. run all the code in .ipynb.

**Note**: 1. When you need to train and test the EasyPortrait_eyes model, please make the variable n_class = 3 in the function portraitnet.MobileNetV2( ) in the files train.py and test.py and test_for_single.ipynb, and keep the variable n_class = 3 when you train the model with other datasets. Otherwise, keep n_class = 2. At the same time, you also need to change the labeling code of dataset.py (lines 201–217), the labeling treatment is not the same when training TEETH and when training EYES. 2. If you want to start training again after interrupting training, change the resume variable (line 472) in the train.py file to True.

## References

[1] Geoffrey Hinton. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 3

[2] Alexander Kapitanov, Karina Kvanchiani, and Kirillova Sofia. Easyportrait - face parsing and portrait segmentation dataset. *arXiv preprint arXiv:2304.13509*, 2023. 2

[3] Laurent Mih. Aisegment.com matting human datasets. https://www.kaggle.com/datasets/laurentmih/aisegmentcom-matting-human-datasets, 2019. Accessed: 2024-10-13. 2

[4] OpenCV Contributors. OpenCV Library on GitHub. https://github.com/opencv/opencv/tree/4.x, 2024. Accessed: 2024-10-13. 4

[5] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 2

[6] Xiaoyong Shen, Aaron Hertzmann, Jiaya Jia, Sylvain Paris, Brian Price, Eli Shechtman, and Ian Sachs. Automatic portrait segmentation for image stylization. In *Computer Graphics Forum*, volume 35, pages 93–102. Wiley Online Library, 2016. 2

[7] Song-Hai Zhang, Xin Dong, Hui Li, Ruilong Li, and Yong-Liang Yang. Portraitnet: Real-time portrait segmentation network for mobile device. *Computers & Graphics*, 80:104–113, 2019. 2, 3