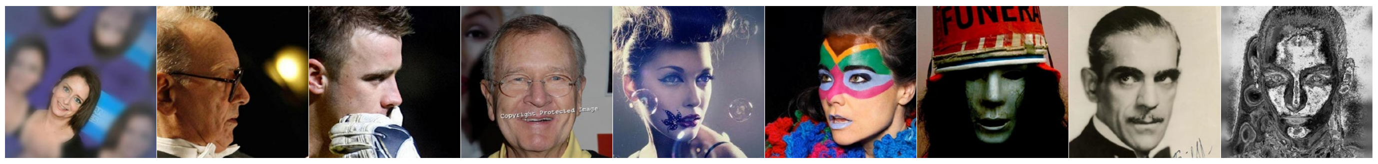
Project I: CelebAMask Face Parsing

Li Hantao, G2101725H, MSAI, hli038@e.ntu.edu.sg

I. Dataset Analysis

This project will use the data selected from the CelebAMask-HQ Dataset [1]. We can only utilize the 5,000 images in the training set and 1,000 in the validation set to train the network. It is crucial to analyze the target dataset for artificial intelligence tasks before we begin implementing the model. The image size in this dataset is 512. The format is a .jpg file with 24-bit color depth and 96 dpi, while the label mask has the format of an 8-bit .png file with the palette.

For the whole dataset, all faces are generally located in the center of the image, with the front or oblique side facing the camera. There is no blatant obstruction on the face except the 'glasses' label. Observing the image shows that the overall image brightness, contrast, and saturation are average, and there are no obvious overexposure, underexposure, and white balance offset.



(a) (b) (c) (d) (e) (f) (g) (h) (i)

However, there are exceptions. In (a), the person's face is inclined and offset, only accounting for a small proportion of the overall image, showing that randomly rotating in data augmentation needs to be considered. In (b) and (c), the person is completely facing the camera laterally. The face in (c), (d), and (e) are covered by glasses. The face in (f) is painted with paint, while even the whole face in (g) is a mask, showing that cutout needs to be considered. The contrast and white balance of (e) and (h) have apparent deviation, and the saturation has a noticeable decrease, indicating that the jittering in contrast, saturation, and hue should be considered. Such strange pictures like (i) in the dataset show that normalization is necessary.

II. Data Augmentation

In this project, the model training pipeline is implemented by the mmsegmentation codebase [2], which has many popular segmentation methods with modular design and provides detailed documentation.

For the circumstance that we only have 5,000 (+1,000) training images to train a semantic segmentation model, it is essential to implement enough skills in the data augmentation pipeline to avoid overfitting as much as possible. Thus, we design a series of ablation studies and contrast experiments to find a better pipeline of data augmentation. However, due to the GPU limit, that is, the computational resource limit, we cannot run full network training many times to test the result of different data augmentation methods. In this project, we only run the baseline model, ResNet50 + DeepLabV3 plus + FCN, for a limit number of iterations (20k) with four as the batch size. The loss function is CE with weight, which will be mentioned in the following section. There is no doubt that we cannot find the authentic trend and enhancement of augmentations in such a small number of epochs with only the baseline model instead of the final big model we choose, but this seems the only reasonable way for this project.

Firstly, we apply color jittering, named photometric distortion in mmcv, to the input image, including adjusting brightness, contrast, saturation, and hue. Each transformation is applied with a probability of 0.5. The default setting of mmcv is brightness delta = 32, hue delta = 18, contrast and saturation range = (0.5, 1.5). Since the images in the dataset do not have such a huge difference, we added an additional set of more moderate parameters, reducing all adjustment ranges to half of the original. The results of the photometric distortion test are shown in Table I, in No.1, No.2, and No.3. The experimental results are consistent with our conjecture that color jittering helps avoid overfitting, but the magnitude should not be too large.

TABLE I

No.	Data Augmentation Type	mIoU	Accuracy (Decoder)	Overfitting*
1	Baseline**	77.15	97.2	Yes
2	Partly Color Jittering***	77.33	97.4	No
3	Color Jittering	77.20	97.2	No
4	Partly Color Jittering + Rotate 15°	77.77	97.1	No
5	**Partly Color Jittering + Rotate 25°**	**77.88**	**96.9**	**No**
6	Partly Color Jittering + Rotate 35°	77.47	96.7	No
7	Partly Color Jittering + Rotate 25° + Small Cutout	77.31	96.8	No
8	Partly Color Jittering + Rotate 25°+ Big Cutout	77.85	96.7	No
9	SWIN-B 5K	79.26	97.9	Yes
10	SWIN-B 10K	79.53	97.6	Yes
11	**SWIN-B 10K + Crop**	**79.59**	**97.6**	**Yes**

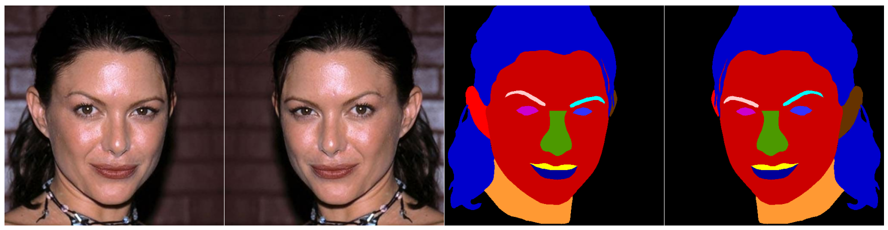
* No overfitting for AI models is impossible. However, the 'overfitting' here means the mIoU decrease before the last epoch while the Accuracy is still increasing. If the model is overfitted before the final epoch, mIoU and Accuracy show the result of the last evaluation before overfitting.

**Baseline is the network without any method of data augmentation.

***Partly Color Jittering is the half ranges of original color jittering, mentioned above.

Secondly, we apply random rotating to the input image, with the probability of 0.5, since not all images are strictly vertical. We tried with three different degrees, from 15° to 35°, as the range of rotating. The results are shown in No.4, No.5, and No.6.

Thirdly, we apple cutout to the input image for there is tiny overlapping in the image, such as a sentence line. We use two different bags of cutout parameters. The 'small' one means implementing one or two cutouts on the size of 32 or 48 with the probability of 0.3, while the 'big' one means the size of 32 or 64 with the probability of 0.5. The results are shown in No.7, and No.8.



Fourthly, we considered the random flip. For the fact that the label has the attribute of left and right, we cannot implement the random flip in the training pipeline, or the model will fail to learn how to figure out the left and right eye. However, 5,000 images are too small to train a robust model, so we wrote a python script to flip them in the correct left and right direction, shown above. In this way, we can double-size our training set to 10,000 images. Because of the non-overfitting in the baseline training, we chose another model, SWIN-B [3], to test the performance, shown in No.9 and No. 10.

Lastly, the random crop is always a critical data augmentation method in the semantic segmentation tasks. However, the average location and size of the face in the image have a certain degree of consistency in this dataset. Thus, we cannot determine whether it is appropriate to implement cropping, so we do another contrast experiment with the random cropping. First, we resize the image's edge from 512 to 512~640, then crop it back to 512. The result has shown in No. 11. After the whole testing procedure, we decided to use the following method:

a) Random Rotate the image with range in 25°, Prob = 0.5.
b) Resize the image's edge from 512 to 512~640, then crop it back to 512.
c) Color Jittering, brightness = 16, hue = 9, contrast and saturation range = (0.75, 1.25), Prob = 0.5.
d) Normalize the image with the mean and std of train + validation set.
e) Double size the training set with flip.

III. Loss Function and Model

For the loss function selection, we tested the cross-entropy loss, focal loss, and dice loss in a similar way. In addition, we tried the weighted loss function for the reason that the small part of the image, such as necklace and earring, a high weight may accelerate the learning on those parts. The weight we set are [0.95, 0.95, 0.95, 1, 1, 1, 1.05, 1.05, 1.05, 1.05, 1, 1, 1, 0.95, 1.05, 1.5, 2, 1, 1.05], corresponding with the label of ['Background', 'Skin', 'Nose', 'Eye glasses', 'Left eye', 'Right eye', 'Left brow', 'Right brow', 'Left ear', 'Right ear', 'Mouth', 'Upper lip', 'Lower lip', 'Hair', 'Hat', 'Ear ring', 'Necklace', 'Neck', 'Cloth']. Moreover, we set another higher weight, which doubled the fluctuation of the weights to determine the appropriate weights in experiments. The results are shown in Table II. With the information we obtained from the table, we decided to utilize weighted cross-entropy and lovasz loss with ratio of 1:2 as the loss function, which is the same as most famous models in semantic segmentation.

TABLE II

No.	Data Augmentation Type	Decode head : Auxiliary head	mIoU	Accuracy (Decoder)	Overfitting
1	Weighted Cross Entropy	1 : 0.4	77.33	97.4	No
2	Weighted Cross Entropy + Focal Loss (1:1)	1 : 0.4	77.10	97.4	No
3	Weighted Cross Entropy + Focal Loss + Dice Loss (1:1:1)	1 : 0.4	77.08	97.2	No
4	Unweighted Cross Entropy	1 : 0.4	76.88	97.3	No
5	High Weighted Cross Entropy	1 : 0.4	77.27	97.3	No
6	Weighted Cross Entropy	1 : 1	76.83	97.2	Yes
7	Weighted Cross Entropy + Lovasz Loss (2:1)	1 : 0.4	78.22	97.5	No
8	Weighted Cross Entropy + Lovasz Loss (1:1)	1 : 0.4	77.92	97.5	No
9	**Weighted Cross Entropy + Lovasz Loss (1:2)**	**1 : 0.4**	**78.40**	**97.5**	**No**

We chose the ConvNeXt-XL as the backbone of our network while utilizing UPerNet as the decoder and FCN as the auxiliary head [4-6], which is the same model arrangement as the basic ConvNeXt's model in the mmsegmentation [7]. ConvNeXt achieves the accuracy of ImageNet's top-1 by convolution structure, which runs counter to the popular trend of using the transformer to solve visual problems in these years. ConvNeXt does not have a particularly complex or innovative structure. Every detail of its network has been adopted in more than one network. Starting from ResNet-50 or ResNet-200 [8], it successively draws lessons from the idea of SWIN from the five perspectives, including macro design, deep separable convolution (ResNeXt [9]), inverse bottleneck layer (MobileNet V2 [10]), large convolution core and detailed design.

TABLE III

No.	Network	mIoU
1	SWIN-B + UPerNet + FCN	79.53
2	SWIN-L + UPerNet + FCN	79.60
3	**ConvNeXt-XL + UPerNet + FCN**	**79.63**
4	MiT b5 + SegFormer [11]	79.50

To decide to use ConvNeXt, we test a series of SOTA models in mmsegmentation, shown in Table III, where ConvNeXt shows the best performance while relatively fast training speed. We do not implement any data augmentation method in this test, except flipping to double size the image, in order to speed up the convergence rate of model.

IV. Result

We complete all the training and testing procedures in the following two platforms in Table IV.

TABLE IV

Platform I	Platform II	Parameters
GPU: NVIDIA GeForce RTX 3090 CPU: AMD EPYC 7642 48-Core Processor Ubuntu~18.04 PyTorch: 1.8.1+cu111	GPU: NVIDIA GeForce RTX 3080 Laptop CPU: AMD Ryzen 9 5900HX Windows 10 PyTorch: 1.8.1+cu111	Flops: 223.34 GFLOPs Params: 391.04 M

The pre-trained models on ImageNet-21k are used [7, 12]. We trained the ConvNeXt-XL with the whole 12,000 images (5,000 + 1,000, doubled) for 160,000 iterations with batch size of four (about 56 epochs). Warn up of 1,500 iterations are utilized. AdamW and poly are used as the optimizer and learning rate settings. More detailed hyper-parameters could be found in the config file. We have no validation set; thus, the curve below only contains the curve of the training procedure and testing procedure, which was obtained by submitting the result to the website. The training curve and testing curve are shown in the figure below.

The loss in figure is the training loss. The shaded area is the range of best result and best result with TTA, which will be discussed in the following section. At 144,000 iterations, we decrease the loss weight, for the reason that the current learning rate and loss seem too large to continue the training procedure. However, the training mIoU increases after that while the testing mIoU does not move.

We implemented a series of testing on the checkpoint of 72k iterations of ConvNeXt to figure out the best method of test-time augmentation (TTA). As we all know, the original size of the image is 512. Here, we only discuss the different sizes of testing instead of flip testing so that the only parameter is the ratio we resize the image. For example, the ratio of [0.75, 1.0, 1.25] means that we use three images with the size of 384, 512, and 640 to generate one output. The results are shown in Table V. Nevertheless, in another iterations' checkpoint, such as 56k or 88k, the TTA method can no longer provide enhancements in testing mIoU. This may indicate that the TTA with different sizes can only improve the network in the specific period. Thus, in the testing curve in the above section, we do not use the TTA.

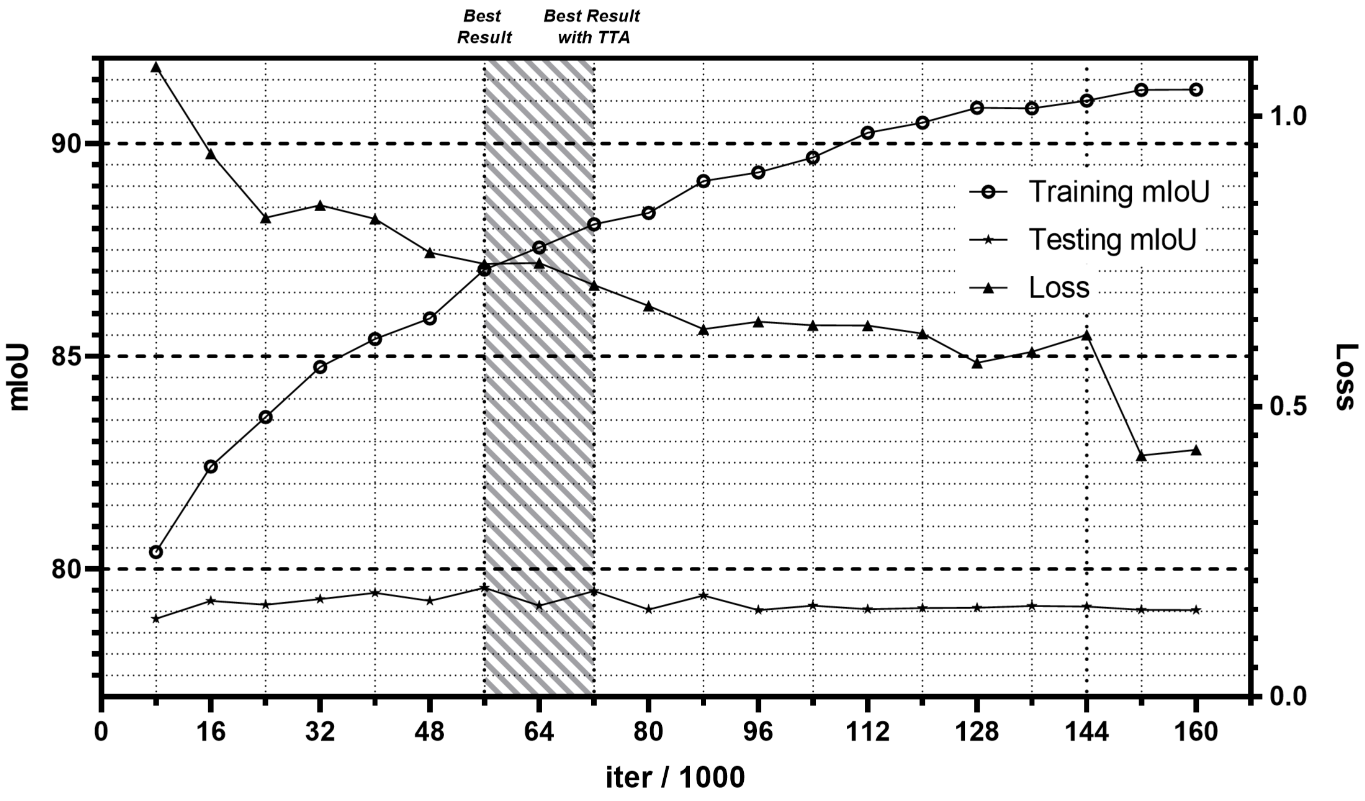


TABLE V

No.	Resize Ratios	Input Image Sizes	mIoU
1	Baseline (1.0)	512	79.480
2	1.0 + 1.25	512 + 640	79.550
3	0.75 + 1.0	384 + 512	79.533
4	0.75 + 1.0 + 1.25	384 + 512 + 640	79.679
5	**0.5 + 0.75 + 1.0 + 1.25 + 1.5**	**256 + 384 + 512 + 640 + 768**	**79.754**
6	1.0 + 1.25 + 1.5	512 + 640 + 768	79.442
7	0.5 + 0.75 + 1.0 + 1.25	256 + 384 + 512 + 640	79.612

V. Conclusion

In this project, we implement a series of ablation studies and contrast experiments to find a better pipeline of data augmentation, a better combination of the loss function, and a better backbone and decoder head of the network. We also try to find the best TTA method; however, it doesn’t work in all checkpoints’ outputs. After the model's training with the optimal parameters we searched, the highest testing mIoU is **79.75417** at 72,000 iterations.

There are still additional methods that can enhance the model's performance because we always utilize the default learning rate and optimizer instead of searching the optimal hyper-parameters of them with the limited computing resource and GPU. Moreover, we can also implement a flip-augmentation in the TTA procedure. We can write a custom flip, which can double-flip the left/right attribute label, similar to the data augmentation. In this way, we believe that the final score will be higher than the current one.

[1] Lee, C. H., Liu, Z., Wu, L., & Luo, P. (2020). Maskgan: Towards diverse and interactive facial image manipulation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 5549-5558).
[2] https://github.com/open-mmlab/mmsegmentation
[3] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., ... & Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 10012-10022).
[4] Liu, Z., Mao, H., Wu, C. Y., Feichtenhofer, C., Darrell, T., & Xie, S. (2022). A ConvNet for the 2020s. arXiv preprint arXiv:2201.03545.
[5] Xiao, T., Liu, Y., Zhou, B., Jiang, Y., & Sun, J. (2018). Unified perceptual parsing for scene understanding. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 418-434).
[6] Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3431-3440).
[7] https://github.com/open-mmlab/mmsegmentation/tree/master/configs/convnext
[8] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
[9] Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1492-1500).
[10] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4510-4520).
[11] Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J. M., & Luo, P. (2021). SegFormer: Simple and efficient design for semantic segmentation with transformers. Advances in Neural Information Processing Systems, 34.
[12] https://download.openmmlab.com/mmclassification/v0/convnext/downstream/convnext-xlarge_3rdparty_in21k_20220301-08aa5ddc.pth