CE 7454 Project 1 Report: CelebAMask-HQ Face Parsing

Xinghe Chen
CCDS, Nanyang Technological University
Nanyang Avenue, Singapore 639798
xinghe001@e.ntu.edu.sg

Abstract

Face parsing is an important task in facial image analysis, involving the segmentation of facial components. In this project, we use an adapted SegFormer model with fewer than two million trainable parameters. The model is applied to a subset of the CelebAMask-HQ dataset containing high-resolution images annotated with 19 facial components. Our model achieves a high mean intersection over union (mIOU) score of ***56.67%****. This demonstrates SegFormer's superior performance compared to traditional CNN-based architectures like U-Net, DeepLabV3Plus, and FastSCNN. We also show optimisation and regularisation techniques improve performance significantly. Additionally, efficiency analysis reveals trade-offs between computational complexity and model efficiency. The challenges of long-tail distribution and the potential domain shift between the training, validation, and test sets are discussed. These emphasise the need for further study in face segmentation beyond the project's scope. Codes available at https://github.com/XChen1998/CE-7454-Project-1*

1. Introduction

The CelebAMask-HQ dataset is a high-quality facial image dataset for facial-related tasks, with a focus on detailed facial attributes and regions [3]. It contains 30,000 high-resolution images, each annotated with 19 facial components, including eyes, nose, mouth, hair, and accessories. The dataset's complexity and rich annotations provide an ideal study object for advancing research in image segmentation and facial analysis. In this course project, we use a subset of the dataset, consisting of 5,000 training samples, 1,000 validation samples, and 100 test samples, tailored for a competition hosted on Codabench.

As the parent task of face segmentation, image segmentation is a fundamental problem in computer vision that has garnered significant research attention. Traditional methods, such as edge detection and clustering algorithms like K-means and watershed, are widely used in earlier approaches. However, with the advancement of deep learning, convolutional neural networks (CNNs) revolutionised the field [4], allowing models of the fully convolutional networks (FCNs) family, including FastSCNN [5], U-Net [6], and DeepLabV3Plus [1] to achieve superior performance by effectively learning spatial hierarchies and feature representations. More recently, transformer-based models have pushed the state-of-the-art further, using self-attention mechanisms to capture both local and global dependencies [2], leading to significant improvements in segmentation tasks [9].

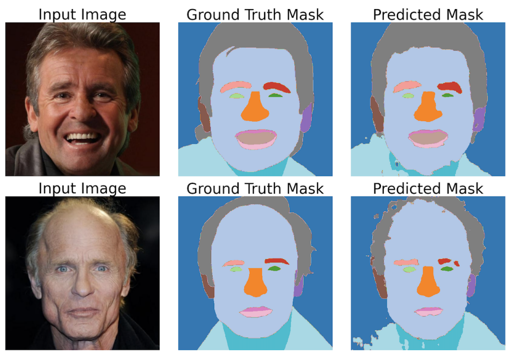


Figure 1. An example of the CelebAMask-HQ face parsing task.

In this project, we utilise SegFormer, a transformer-based model, to perform face parsing on the CelebAMask-HQ dataset. An example of the task is shown in Fig. 1. The primary objective is to develop a model to generate segmentation masks from facial images accurately. Our results demonstrate that:

- **Superior performance**: SegFormer outperformed other models, including U-Net, DeepLabV3Plus, and FastSCNN, achieving a competitive mean intersection over union (mIOU) score of **56.67%**.
- **Effective optimisation and regularisation**: We implemented comprehensive optimisation and regularisation techniques, significantly enhancing model performance. These methods are discussed in detail in Section 2.2.
- **Runtime and complexity trade-offs**: the runtime anal-

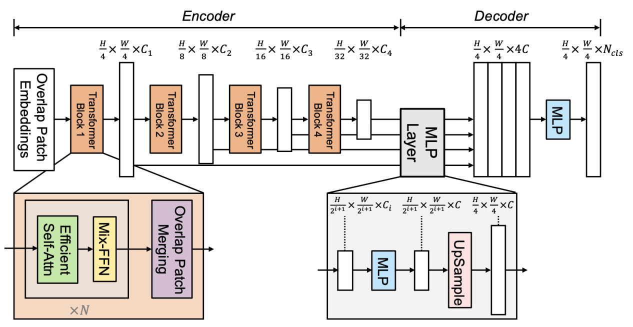
ysis demonstrates a trade-off between performance and computational complexity. Even when models have a similar number of parameters, their computational demands can differ substantially, as discussed in our findings.

2. Method

This section outlines the overall approach in this project, including the model architecture, optimisation techniques, and regularisation techniques used to achieve optimal performance. It discusses the model's design, the strategies used to enhance convergence during training, and the regularisation techniques applied to mitigate overfitting and improve the model's generalisability.

2.1. Model Architecture

In this project, we utilise SegFormer to address the face parsing task, leveraging its transformer-based architecture for semantic segmentation. As shown in Fig. 2, SegFormer comprises two main components: a hierarchical transformer encoder and an all-MLP decoder. The encoder captures multi-scale features at various resolutions and effectively integrates both coarse and fine details, which is crucial for accurate dense predictions. The decoder efficiently fuses these features to generate the final segmentation.



Encoder
Decoder
$\frac{H}{4}\times\frac{W}{4}\times C_1$
$\frac{H}{8}\times\frac{W}{8}\times C_2$
$\frac{H}{16}\times\frac{W}{16}\times C_3$
$\frac{H}{32}\times\frac{W}{32}\times C_4$
$\frac{H}{4}\times\frac{W}{4}\times 4C$
$\frac{H}{4}\times\frac{W}{4}\times N_{cls}$
Overlap Patch Embeddings
Transformer Block 1
Transformer Block 2
Transformer Block 3
Transformer Block 4
MLP Layer
MLP
Efficient Self-Attn
Mix-FFN
Overlap Patch Merging
$\times N$
$\frac{H}{2^{i+1}}\times\frac{W}{2^{i+1}}\times C_i$
$\frac{H}{2^{i+1}}\times\frac{W}{2^{i+1}}\times C$
$\frac{H}{4}\times\frac{W}{4}\times C$
MLP
UpSample

Figure 2. The original SegFormer architecture in [9].

Unlike other vision transformers (ViTs) that produce single-resolution features, SegFormer employs a hierarchical structure with overlapping patch merging, enhancing both efficiency and scalability. This design resembles that of FCNs such as U-Net. However, while FCNs typically rely on modules like atrous spatial pyramid pooling (ASPP) to capture global contextual information, SegFormer inherently captures non-local context through its attention mechanism. This enables SegFormer to achieve a larger effective receptive field and improves its segmentation performance.

2.2. Optimization and Regularization

Although the SegFormer model is powerful, training transformer-based models from scratch is challenging due to their high data and computational requirements. In this course project, however, the available training dataset is small, and computational resources are limited. To address these issues and maximise the model's potential, a series of optimisation and regularisation techniques are applied to make the training process more effective.

2.2.1 Data Whitening

To standardise the input data, we compute the channel-wise mean (μ_c) and standard deviation (σ_c) across all images in the training dataset. These values are then used to normalise each image by applying the transformation $\mathbf{x} = (\mathbf{x} - \mu)/\sigma$. This whitening process ensures numerical stability and uniformity across the RGB channels, thereby improving model convergence.

2.2.2 Weighted Loss Function

For the multi-class segmentation face parsing task, the cross-entropy loss is widely used. Given that the true label $y_c = 1$ if the sample belongs to class c, and the predicted probability for class c is $\hat{y}_c$, the cross-entropy loss is defined as:

$$\mathcal{L}_{\text{CE}} = -\sum_{c=1}^{C} y_c \log(\hat{y}_c), \tag{1}$$

where C is the total number of classes.

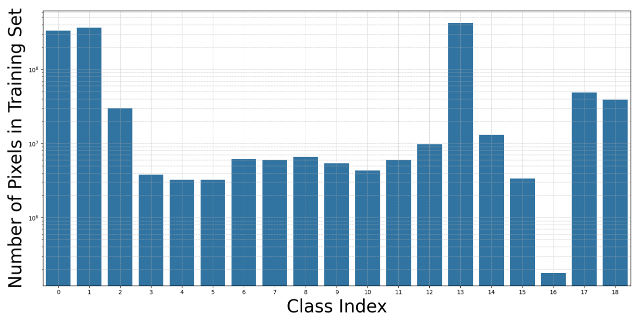


Figure 3. Class frequency distribution in the training set.

However, as shown in Fig. 3, the dataset shows a long-tail distribution, with some labels appearing much less frequently than others. For example, class 16's frequency is at least two orders of magnitude lower than classes 0, 1, and 13. To address this class imbalance, the loss function can be weighted by assigning each class a weight w_c. These weights are computed as the inverse of the log-transformed class frequency, normalised by the total sum of all weights. The weight for each class is given by:

$$w_c = \frac{1}{\log(f_c + \epsilon)} \cdot \frac{1}{\sum_{i=1}^{C} \frac{1}{\log(f_i+\epsilon)}}, \tag{2}$$

where f_c is the frequency of class c, and ϵ is a small constant added for numerical stability. The weighting formula in Eqn.2 is then applied to Eqn.1 to address the class imbalance in the dataset, resulting in the weighted loss shown in Eqn. 3.

$$\mathcal{L}_{\text{weighted}} = -\sum_{c=1}^{C} w_c \cdot y_c \log(\hat{y}_c). \tag{3}$$

The inverse logarithm of class frequency is used because directly inverting the class frequency would assign extremely large weights to less frequent labels. These overly large weights can confuse the model and lead the optimisation process to diverge.

2.2.3 Dynamic Optimiser Pipeline

In this project, we implemented a dynamic optimiser pipeline to enhance both optimisation and regularisation throughout model training.

As shown in Algorithm 1, we start with the Adam optimiser, using an initial learning rate l_{initial} and early stopping based on validation loss. This phase allows the model to converge quickly due to the efficiency and adaptability of the Adam optimiser. Once the patience threshold is reached, we switch to the SGD optimiser with an initial learning rate $l_{\text{SGD\_initial}}$. The learning rate is halved each time patience is exceeded, continuing until it falls below a predefined threshold, $l_{\text{SGD\_min}}$, or the maximum number of epochs is reached. After each patience event, the best weights are reloaded. The SGD phase is designed to help the model escape local minima, as the stochastic nature of SGD introduces implicit regularisation.

In this algorithm, the behaviour of the scheduler is introduced in Section 2.2.4.

Algorithm 1 Dynamic optimiser pipeline

```
 1: Goal: Optimise parameters, $\theta$, of the model
 2: Initialise: Adam optimiser with learning rate $l_{\text{initial}}$,
    SGD with learning rate $l_{\text{SGD\_initial}}$, patience $n$, valida-
    tion loss $\mathcal{L}_{\text{val}}$, threshold $l_{\text{SGD\_min}}$, scheduler
 3: while epoch $\leq$ max number of epochs do
 4:     Update $\theta$ via Adam to minimise $\mathcal{L}_{\text{val}}$
 5:     Scheduler step
 6:     if no improvement in $\mathcal{L}_{\text{val}}$ for $n$ steps then
 7:         Load best $\theta^*$ and switch to SGD
 8:         Set $l_{\text{SGD}} = l_{\text{SGD\_initial}}$
 9:         Reset scheduler
10:         while $l_{\text{SGD}} \geq l_{\text{SGD\_min}}$ and epoch $\leq$ max number
    of epochs do
11:             Update $\theta$ via SGD to minimise $\mathcal{L}_{\text{val}}$
12:             Scheduler step
13:             if no improvement in $\mathcal{L}_{\text{val}}$ for $n$ steps then
14:                 Load best $\theta^*$ and halve $l_{\text{SGD}}$
15:                 Reset scheduler
16:             end if
17:         end while
18:     end if
19:     Load final best $\theta^*$
20: end while
```

2.2.4 Learning Rate Scheduler

In addition to the dynamic optimiser strategy, we also include cosine annealing to improve the optimisation process. This learning rate scheduler progressively decreases the learning rate using a cosine function. This method provides a smoother transition from a higher learning rate to a lower one over the course of training, helping to avoid overshooting in optimisation. The learning rate starts from a maximum value and gradually reduces to a minimum. The cosine annealing schedule for the learning rate $l(t)$ at time step t is given by:

$$l(t) = l_{\text{min}} + \frac{1}{2}(l_{\text{max}} - l_{\text{min}}) \left(1 + \cos\left(\frac{t}{T}\pi\right)\right) \tag{4}$$

where l_{max} is the maximum learning rate, l_{min} is the minimum learning rate, T is the total number of training steps, and t is the current time step. In this project, the l_{min} is set to $\frac{1}{2}l_{\text{max}}$, with l_{max} being the initial learning rate for each phase, and t is reset to 0 after the patience is reached.

2.2.5 Data Augmentation

Since this project provides a training set of only 5,000 samples, various data augmentation techniques are applied to regularise the model, thereby preventing overfitting and improving the model's generalizability. These techniques include random cropping and re-scaling of images, as well as random adjustments to exposure, contrast, saturation, and sharpness. Additionally, random Gaussian noise is added to the training samples to further augment the data.

However, given that the face parsing task requires the model to distinguish between specific facial components such as the left and right eyes, augmentations that break the spatial relationships of facial features are avoided. As a result, transformations like random rotation, horizontal flip, and vertical flip are not applied in this augmentation process.

Model	**Val. mIOU (↑)**	**Val. F-measure (↑)**	**Test mIOU (↑)**	**Parameters (↓)**	**Runtime (s/epoch) (↓)**
U-Net	66.68%	79.98%	53.09%	1,941,411	228
DeepLabV3Plus	67.22%	80.38%	53.30%	1,964,394	328
FastSCNN	70.62%	82.77%	54.74%	<u>1,154,435</u>	<u>171</u>
SegFormer	<u>71.38%</u>	<u>83.34%</u>	<u>56.67%</u>	1,975,683	871

Table 1. Comparison of models based on validation and test mIOU, validation F-measure, number of parameters, and runtime measured in seconds per epoch. ↑ indicates higher is better, and ↓ indicates lower is better. The best results are highlighted with underline.

3. Experiments

This section details the experimental setup and presents the main results of this project.

3.1. Experimental Setup

All experiments are conducted on a server with a 24-core CPU, 64 GB of system RAM, and an Nvidia RTX 4090 GPU. The implementation is carried out using PyTorch.

For detailed hyperparameter settings, please refer to the interactive Jupyter notebooks for different models. The random seed is fixed at 42 for all experiments to ensure reproducibility.

3.2. Convergence Analysis

As shown in Fig. 4, the optimisation process benefits from the dynamic optimiser pipeline introduced in Section 2.2.4. Each optimiser switch helps mitigate overfitting, enabling the model to capture more fine-grained patterns from the limited amount of data. Additionally, the initial phase of the Adam optimiser allows for rapid model warm-up, reducing computational costs.

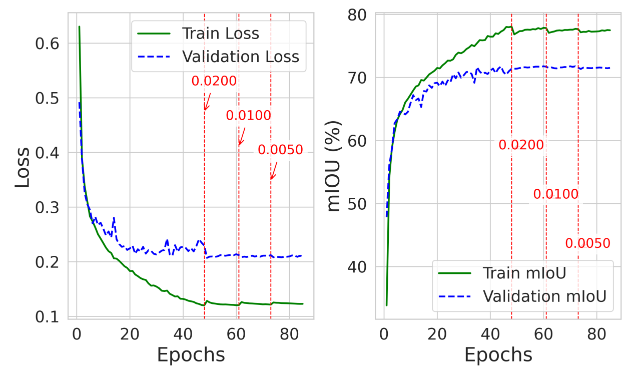


Figure 4. Convergence plot of the dynamic optimiser pipeline. The red lines indicate optimiser switches, with the numbers representing the corresponding learning rates.

3.3. Performance and Efficiency Analysis

To assess the effectiveness of the SegFormer model, we compare its performance against three baseline models:

- **U-Net**: selected for its popularity and its simple yet highly effective hierarchical design for segmentation tasks [6].
- **DeepLabV3Plus**: selected for its advanced architecture, which incorporates an ASPP module to capture multi-scale features [1].
- **FastSCNN**: selected for its lightweight architecture, offering a strong balance between computational efficiency and segmentation performance [5].

In this project, the mIOU, calculated as $\frac{1}{N}\sum_{i=1}^{N}\frac{|A_i \cap B_i|}{|A_i \cup B_i|}$, where A_i and B_i represent the predicted and ground truth regions for class i, and the F-measure, calculated as $2 \times \frac{\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$, are used for model evaluation.

Table 1 summarises the main results. The SegFormer model achieves the highest validation mIOU (71.38%), demonstrating its superior capability in this segmentation task. This further validates the efficacy of attention mechanisms, reinforcing that even in the context of image segmentation, attention is all you need [7]. In contrast, the U-Net model shows the worst performance, with validation mIOU being 66.68%, likely due to its limited ability to capture long-range dependencies. Although DeepLabV3Plus has an ASPP module for extracting large-scale spatial features, its performance (67.72%) does not meet expectations, possibly due to the constraint on the number of parameters, which forces the use of a smaller, less effective CNN architecture as its base. Notably, FastSCNN, with fewer than 1.2 million parameters, achieves a significant speedup of 2.5 to 5 times faster [1] than the other models at the cost of only a marginal decrease in mIOU to 70.62%. This highlights FastSCNN's powerful design for resource-constrained scenarios. The validation F-measure and test mIOU exhibit similar trends.

Based on the investigation of the class-wise mIOU[2], the performance on long-tail classes, such as class 16, is suboptimal. Without applying weighted loss, the mIOU for these classes can occasionally drop to 0%. Even when the weighted loss is applied, this score improves only

[1]Note that runtime analysis may vary depending on computational resources. The setup used for this project is detailed in Section 3.1.

[2]Class-wise mIOU results can be obtained by running the provided code. The submitted files include some examples.

marginally, typically remaining below 6%. Addressing the challenge posed by long-tail distributions remains an open question in image segmentation [8]. In addition to the long-tail issue, there is a noticeable gap between the validation mIOU and test mIOU. This suggests a potential domain shift between the validation and test sets, or the presence of very hard samples in the test set. However, further investigation into this phenomenon is intricate without access to the test set masks.

4. Conclusion

In conclusion, SegFormer demonstrated superior performance in this face parsing task even with a small dataset, surpassing other models with a competitive mIOU score of 56.67%. Through comprehensive optimisation and regularisation, the model achieved enhanced accuracy, while runtime analysis against the FastSCNN model highlighted the trade-offs between performance and computational complexity. The weighted loss function helps mitigate the long-tail distribution issue, though there is space for further improvement. Future work could investigate the potential domain shift between the training, validation, and test sets to enhance the model's generalizability.

5. Appendix

As a disclosure, the author discussed with Yihua Hu for this project.

References

[1] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018. 1, 4

[2] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1

[3] Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. Maskgan: Towards diverse and interactive facial image manipulation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1

[4] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 33(12):6999–7019, 2021. 1

[5] Rudra PK Poudel, Stephan Liwicki, and Roberto Cipolla. Fast-scnn: Fast semantic segmentation network. *arXiv preprint arXiv:1902.04502*, 2019. 1, 4

[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015. 1, 4

[7] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017. 4

[8] Jiaqi Wang, Wenwei Zhang, Yuhang Zang, Yuhang Cao, Jiangmiao Pang, Tao Gong, Kai Chen, Ziwei Liu, Chen Change Loy, and Dahua Lin. Seesaw loss for long-tailed instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9695–9704, 2021. 5

[9] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in neural information processing systems*, 34:12077–12090, 2021. 1, 2