Project proposal: Clothing segmentation

December 2022

Hongtao Yang, Jiayi Song

Github link:https://github.com/Jayi-S/COMP576-Finalproject

Contents

1. Abstract

Clothing segmentation has growing importance in many aspects and several works have been conducted on it. However, these works focus on designing a well-performed model and lack of study in loss function. Besides, clothing segmentation differs from other semantic segmentation tasks due to the fact that the clothes may vary a lot in appearance. A deep study into loss function is effective and meaningful. In our project, we apply UNet to realize clothing segmentation and conduct several experiments to find the best loss function and function combination in clothing segmentation tasks. Our experiments show that mean IoU loss performs best. Our findings benefit the study of clothing segmentation and inspire other semantic segmentation with large appearance variants.

2. Background/ Motivation

Clothing Segmentation, which means segmenting clothes from pictures and classifying clothes into different categories, has growing influences in many areas. Automatic segmentation of clothes can improve work efficiency of art designers. Besides, for online shopping, it is an essential part of virtual try-on. For virtual try-on, e-commerce companies need to get the accurate contours of their products and customers' wearing. In short, clothing segmentation has a huge economic effect.

The most common method used to solve clothing segmentation problems is semantic segmentation. Semantic segmentation means clustering parts of an image together which belong to the same object class. There have been multiple semantic segmentation methods based on CNN models, like UNet [1] and DeepLabV3 [2], or based on transformer architectures, like segFormer[3].

These models achieve good performance in common semantic segmentation tasks. But due to some features of clothing segmentation, they may not perform well. Firstly, the appearances of clothes vary a lot in many aspects, including size, colors and patterns. Two trousers belong to the same class, but one may be pure yellow and the other may be plaid striped, which is quite different. Secondly, since the segmentation of garment parts is also needed, there are multiple classes. However, since the data of some classes is relatively rare, it's difficult to guarantee accuracy of each class. Finally, the appearance of clothes will be affected by human pose and occlusion.

Due to these difficulties, only studying model structure may not be enough to get a good performance. A deep study into finding the best loss function and function combination will help increase clothing segmentation performance.

3. Broader Scope and Goal

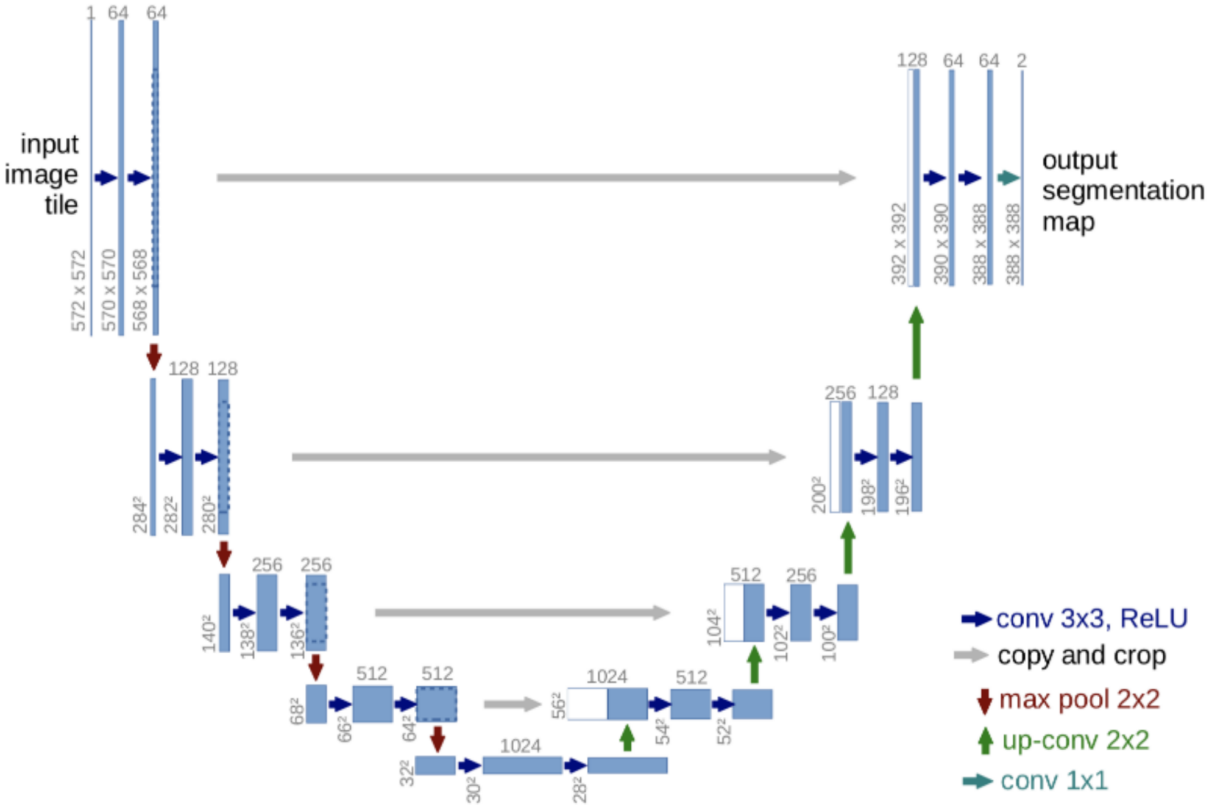
There have been many works in semantic segmentation. Some researchers designed awesome model structures [1-3] while others designed loss functions to improve performance, like focal loss [4], dice loss [5] and Generalized Dice loss [6]. It shows the loss function is also important for semantic segmentation.

In the specific clothing segmentation problem, most works focus on using well-designed models to get good segmentation performance. In [7], authors apply Feature Pyramid Networks in clothing segmentation tasks and outperform other methods. In [8], authors use deformable convolutions to extract the non-rigid geometry-aware features for clothing images. However, less attention is paid to finding a useful loss function in clothing segmentation tasks.

We use UNet to do clothing segmentation. Different from previous works focusing on designing models, we pay attention to loss function in clothing segmentation. We conduct experiments to find the most suitable loss function in this project.

4. Proposed solution and major contributions

4.1 Model structure

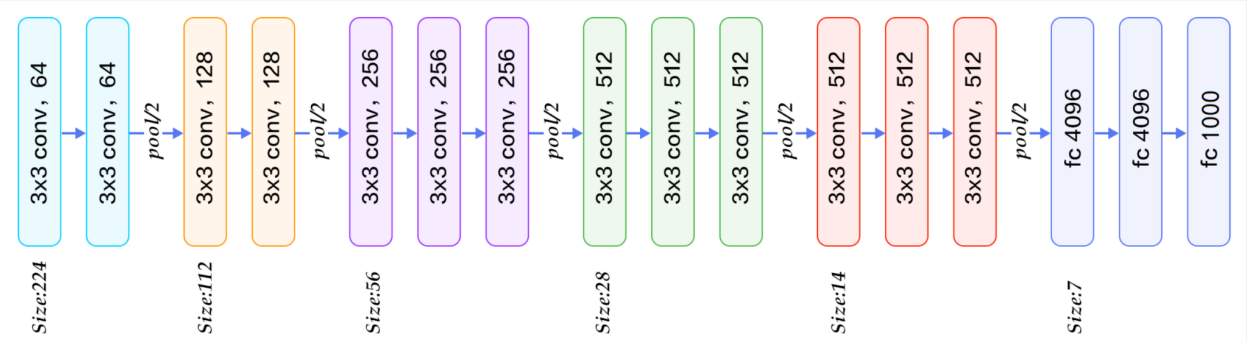


Pic 1. UNet structure

Pic 1 shows the structure of UNet. It has an encoder part and decoder part. The encoder part is composed of convolutional layers and pooling layers. It takes images as input and outputs several large-scale and small-scale features. The decoder part takes in the features extracted by the encoder part. It uses stacked convolutional layers to decode the feature to get the mask result. It uses up-sampling layers to enlarge the mask size. Since large-scale features keep the details of mask and small-scale features focus on objects. UNet uses skip connections to merge large-scale features with up-sampled decoded features. UNet uses both large-scale and small-scale features, so it can output accurate masks.

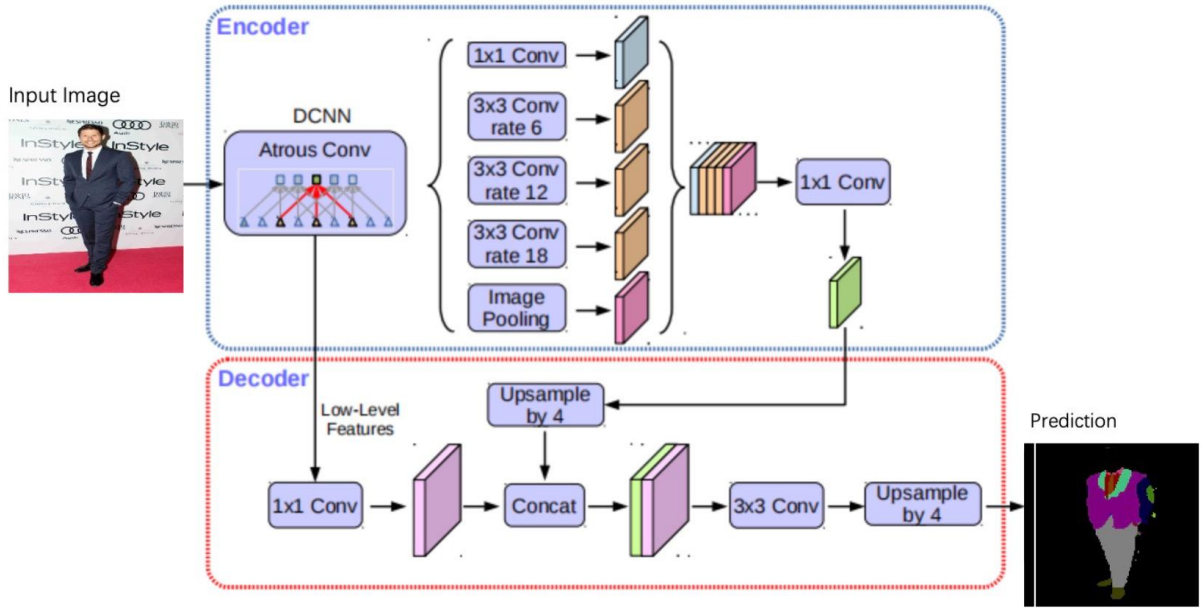
In our experiment, we use VGG16 (Pic 2)[10] as a feature encoder. It uses a 3x3 size convolutional kernel and pooling layer to produce multi-scale features. It can produce 1/2, 1/4,

1/8, 1/16 and 1/32 to input image size. This well-designed structure makes it an ideal baseline for our task.



Pic 2. VGG16 structure

After completing the above experiments, we also tried to build deeplabV3+ to perform the task of clothes segmentation.



Pic 3.deeplabV3+ structure

pic3 is a deeplabv3+ model. For the encoder part, after the image enters the backbone network, two feature layers are obtained, and the shallow feature layer is directly entered into the decoder for 1*1 convolution for channel compression. The purpose is to reduce the proportion of low-level layers. The deep features enter the ASPP module in the encoder. In ASPP module, it includes 1 1*1 convolution, 3 3*3 Atrous convolution with a ratio of 6, 12, and 18, and a global

image Pooling Operation, these operations are followed by a 1*1 convolution. For the decoder part, directly upsample the output of the encoder by 4 times to make its resolution consistent with the low-level feature. After connecting the two feature layers, perform a 3×3 convolution (thinning effect), and then upsample again to obtain a pixel-level prediction.

We use Resnet50 as an encoder. It uses residual connection to make the network deeper, thus producing robust features. Since deeplab v3+ needs a shadow feature and a deep feature, we use the output layer conv2 and conv5.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

4.2 loss functions

There have been several loss functions designed for semantic segmentation tasks. But for a specific challenging task: clothing segmentation, which loss is the most suitable? Limited work focuses on this problem.

We explore different loss functions in this task to find a most useful one. Since some losses may just work well with one specific model, experiments will be conducted based on different models to confirm the final situation.

The first kind of loss is pixel level loss. In clothing segmentation tasks, each pixel can be viewed as a sample. The task is to classify each pixel into the accurate class. Thus, we can use traditional image classification loss to optimize our model, including Cross Entropy loss and Focal loss.

Suppose the predicting result is a one-hot label.

Cross Entropy:

$$\text{Loss} = -\sum_{i=0}^{C-1} y_i \log\ (p_i)$$

Focal loss:

$$\text{Loss} = -\sum_{i=0}^{C-1} \alpha_i (1-p_i)^{\gamma} y_i \log\ (p_i)$$

Derivation of Cross Enrtopy:

$$\frac{\partial \text{CE}}{\partial z_i} = -\frac{y'_i}{y_i} \cdot (y_i - y_i \cdot y_i) + \sum_{t=1, t\neq i}^{n} -\frac{y'_t}{y_t} \cdot ((-y_t \cdot y_i)$$

$$= y_i \cdot y'_i - y'_i + \sum_{t=1, t\neq i}^{n} y_i \cdot y'_t$$

$$= -y'_i + \sum_{t=1}^{n} y_i \cdot y'_t$$

$$= -y'_i + y_i$$

Derivation of Focal loss:

$$\frac{\partial FL}{\partial p} = +y\gamma(1-p)^{\gamma-1} log(p) - y(1-p)^{\gamma}\frac{1}{p} - (1-y)\gamma p^{y-1} log(1-p) + (1-y)p^{\gamma}\frac{1}{1-p}$$

In focal loss, and are hyperparameters. is to adjust the influence of different classes. is to enhance the influence of hard-to-classify samples.

The other kind of loss is image level loss. It takes the input image as a whole and uses set theory to get the loss value.

For any label image Y and output X, calculate the intersection ratio to get the loss

Dice loss:

$$\text{Loss} = 1 - \frac{2|X \cap Y|}{|X| + |Y|}$$

Derivation of Dice loss:

$$\frac{dL_{dice}}{dy} = -\frac{2t(t+y+\varepsilon)-2ty-\varepsilon}{(t+y+\varepsilon)^2} = \begin{cases} \frac{\varepsilon}{(y+\varepsilon)^2} & \text{t=0} \\ -\frac{2+\varepsilon}{(1+y+\varepsilon)^2} & \text{t=1} \end{cases}$$

$$\frac{dL_{dice}}{dx} = \frac{dL_{dice}}{dy}\frac{dy}{dx}$$

:

Since the output value of the model is larger than 0 and smaller than 1, the intersection or union can be calculated directly. We need to make some adjustments to make the loss computable and differentiable. The computable format is:

$$\text{Loss} = 1 - \frac{1}{C}\sum_{i=0}^{C-1} \frac{2\sum_{pixels} y_i p_i}{\sum_{pixels}(y_i + p_i)}$$

Miou loss:

$$\text{Loss} = 1 - \frac{|X \cap Y|}{|X \cup Y|}$$

the computable format is:

$$\text{Loss} = -\frac{1}{C}\sum_{i=0}^{C-1} \frac{\sum_{pixels} y_i p_i}{\sum_{pixels}(y_i + p_i - y_i p_i)}$$

Derivation of Miou loss:

$$\begin{aligned}\frac{\partial \mathcal{L}_{\mathrm{MIOU}}}{\partial P_i} &= -\frac{1}{|C|}\sum_c \frac{\partial \mathcal{L}_{\mathrm{MIOU}}}{\partial P_{ic}} \\ &= -\frac{1}{|C|}\sum_c \frac{\partial\left[\frac{I(P_c)}{U(P_c)}\right]}{\partial P_{ic}} \\ &= -\frac{1}{|C|}\sum_c \left\{\frac{p_{ic}^* \cdot U\left(P_c\right) - I\left(P_c\right)\left(1-p_{ic}^*\right)}{\left[U\left(P_c\right)\right]^2}\right\}\end{aligned} \tag{4}$$

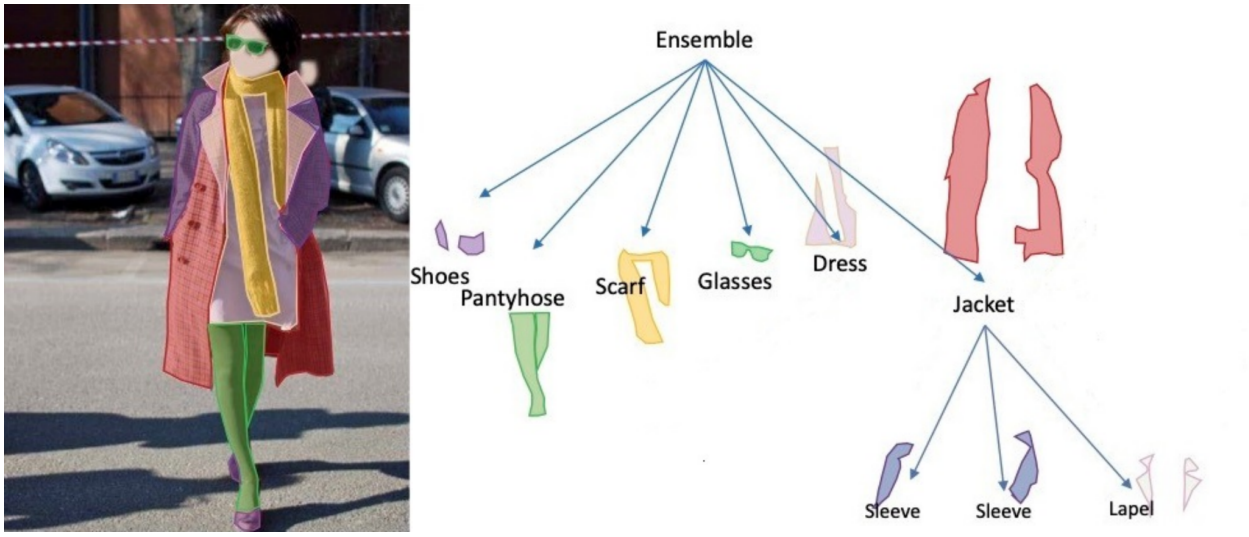
In terms of the classification decision principle, $i \in V$ must belong to a category. Therefore, p_{ic}^* can only correspond to 1 on the true category. (4) can be further simplified as follows:

$$\frac{\partial \mathcal{L}_{\mathrm{MIOU}}}{\partial P_i} = -\frac{1}{|C|}\cdot\frac{1}{U(P_c)} \quad \begin{array}{l} p_{ic}^* = 1, \\ p_{ik}^*\left(k \neq c\right) = 0 \end{array} \tag{5}$$

5. Proposed experiments-datasets, tasks, architectures, expected results

5.1 dataset

In our experiments, we will use the iMaterialist dataset [9]. It provided a large number of images and corresponding fashion/apparel segmentations. It contains images of people wearing a variety of clothing types in a variety of poses. Besides, it has fine-grained class segmentations. All these characters make this dataset suitable for our experiment.



Pic 3. iMaterialist dataset sample

5.2 Experiment settings

Due to the computation limitations and lots of experiments, we don't use all the data. Instead, we use part of the data to ensure all experiments are conducted. Besides, since our goal is not to get the highest performance on this dataset. We don't use large models to improve the performance. Instead, we use lightweight models in our experiments, which can ensure us pay most attention to loss function experiments.

We train our segmentation model with different loss functions. By comparing the results, we can find the most suitable loss and loss combination. We analyze the result with the principle of loss function to describe why the function works or not.

Table 1. Experiment parameters

parameters	value
Input size	256 x 256
Training samples	5000
Validation samples	1000
Learning rate	0.001
Batch size	16
Epochs	100

We train our model on 47 classes(46 clothing classes + background). We conducted three kinds of testing.

Testing 1: 2 classes testing. To test the performance in classifying clothes and background.

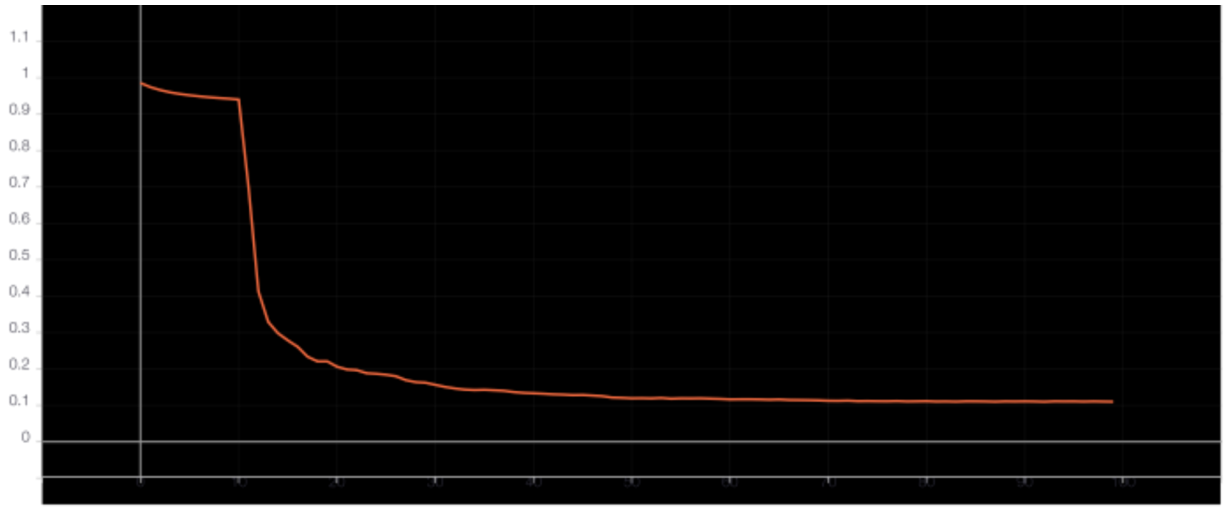
Testing 2(most concerned): 17 common clothes classes testing. To test the performance in segmenting common clothes.

Testing 3: all 47 classes testing. To test the performance in segmenting all types of clothes, including some scarce types, like epaulet and cape.

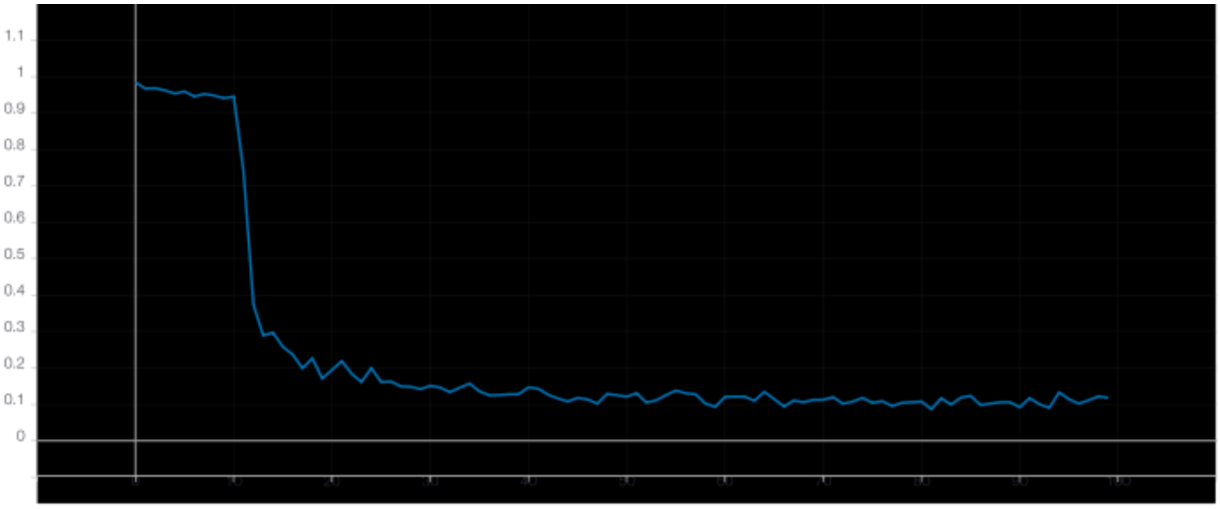
Our evaluation metric means iou metric. It means iou metric. Its value is obtained by averaging intersection over union on all classes.

Please see the specific classes classification in the Appendix 5&6. We have recorded the loss of training 47-classes in different loss functions in the Appendix.

5.3 Experiments results



Pic 3. Training loss chart



Pic 4. Validation loss chart sample

Table 1. Unet Experiment results

	2classes	17classes	47classes
Cross Entropy loss	**0.8791**	0.3103	0.1265
Mean IoU loss	0.8745	**0.4145**	**0.2022**
Dice loss	0.6778	0.3315	0.1457
Dice+CE loss	0.8688	0.2837	0.1177
Focal loss (=0.25, =2)	0.8664	0.3153	0.1298
Focal loss (=0.25, =5)	0.8366	0.2766	0.1096

Table 2. DeepLabV3+ Experiment results

	2classes	17classes	47classes
Cross Entropy loss	0.8886	0.3439	0.1450
MIoU loss	**0.8980**	**0.3791**	**0.1734**
Dice loss	0.7536	0.1384	0.0501
Dice+CE loss	0.8848	0.3399	0.1416

Focal loss (=0.25, =2)	0.8592	0.2976	0.1199
Focal loss (=0.25, =5)	0.8612	0.3040	0.1237

Chart1. Mean IoU Loss

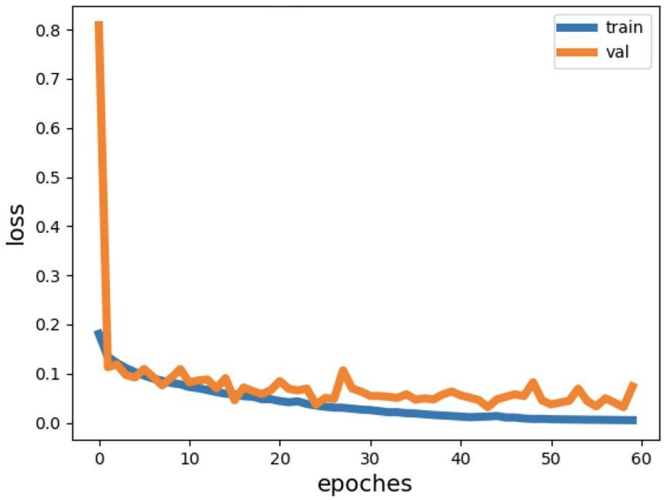


Chart2.Dice Loss

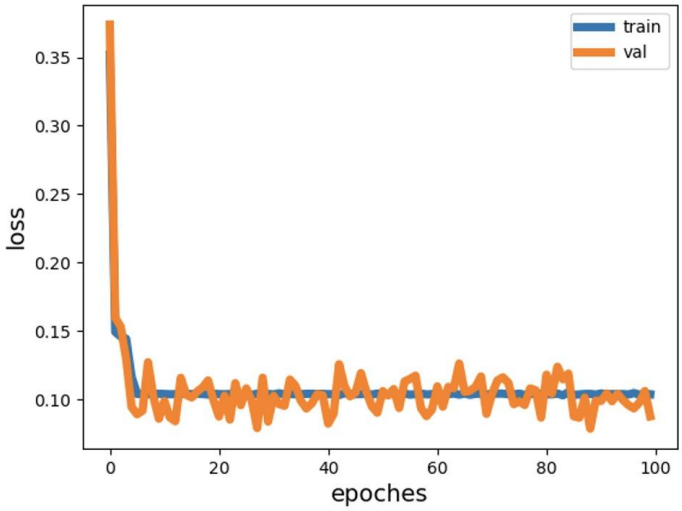


Chart3.Dice & CE loss

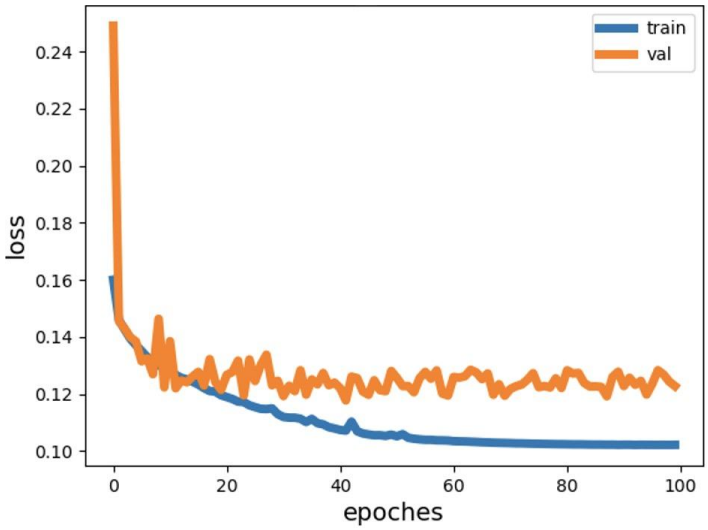


Chart4.CE loss

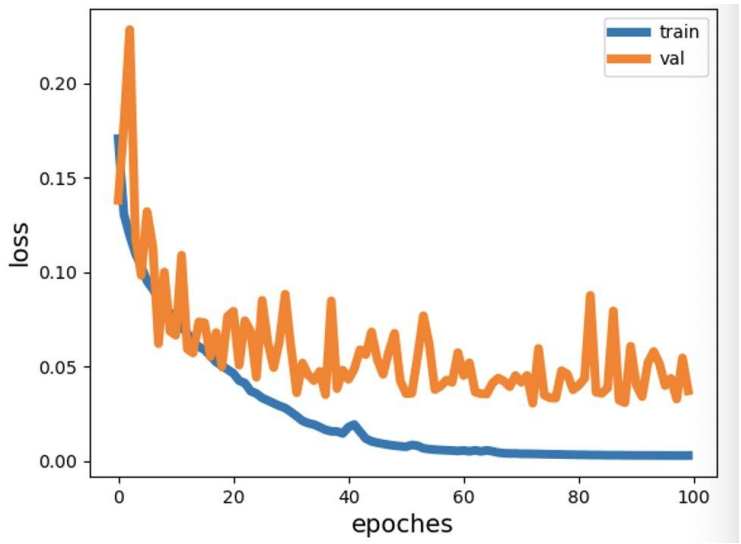


Chart5. focal loss

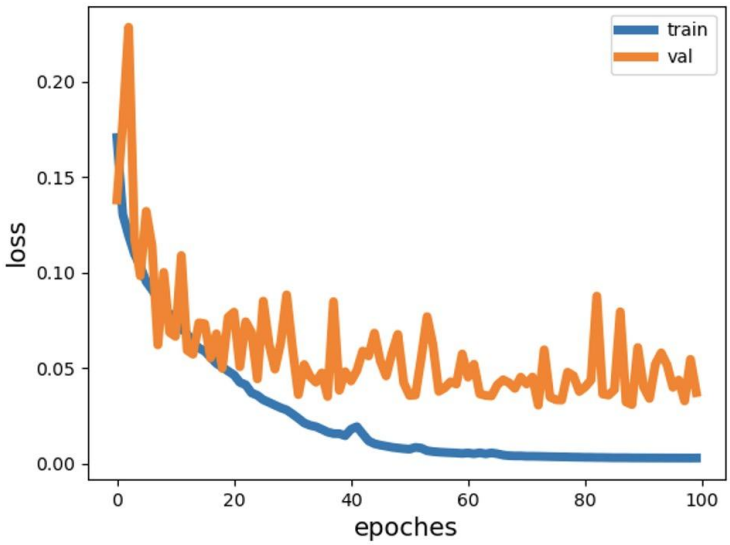
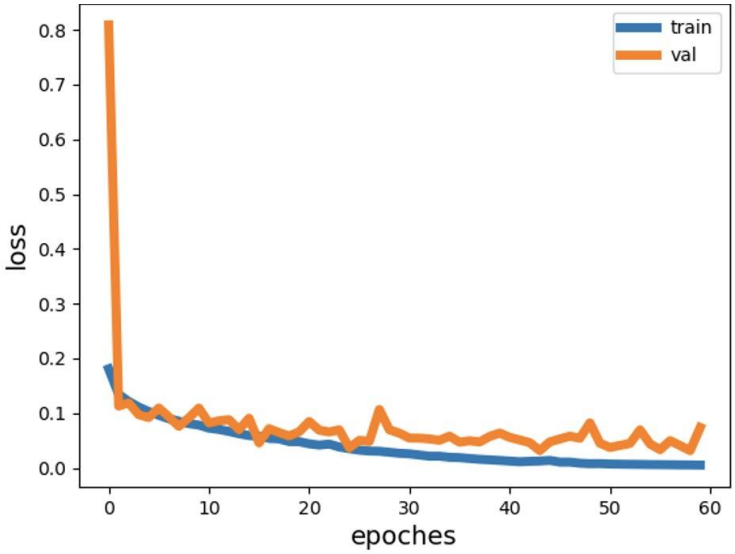


Chart6. Focal_gamma5 loss



5.4 Results Analysis

Pic 3 and pic 4 shows the loss value on training and validation data during the optimization process. We can see that the loss value decreases with almost the same trend. It indicates that our model is optimized gradually with no overfitting happening. That makes our results convincing.

Table 1 shows the mean Iou value of different experiment settings. We can find the value in column 2 is much higher than those in column 3 and column 4. This indicates that classifying a model into 2 classes is much easier than into multiple classes. Since semantic segmentation takes each pixel as a sample, the feature of each sample is affected by neighboring pixels. Low-level features like texture and color are enough to classify clothes and background.

Comparing row 4 and row 5 in table 1&2.(Chart4 and Chart3) We can find that when only using dice loss, the result is unstable. But when combining dice loss with ce loss, the result is relatively acceptable.

In row 3 in table 1 & 2, we can find mean Iou loss performs better than other losses, especially in 17-classes and 47-classes tasks. We consider it's because the mean iou loss takes the whole image to get loss value. Thus, it can grasp more global information.

By comparing column 3 and column 4 in table 1&2. We can find it when we just do 17-classes segmentation. The result is better than doing 47-classes segmentation. It indicates that there are some classes hard to classify. This problem is caused by lacking samples of these classes.

For the two models, We provide both UNet and deeplab v3+ visualization results (in 5.6 Visualization results); On the other hand, We found Unet training speed is relatively fast. In the case of limited hardware resources, Unet can try more parameters changed test quickly;

5.5 Conclusions

1. When we just segment clothes and background, results will be much higher than multiple clothing segmentation. It shows that different clothing types share very similar characters.
2. Only using dice loss will make the training process unstable, causing the result worse than using losses. Instead, it should work with CE loss.
3. Mean IoU loss is found to work better than other losses, since it can make the model get more global features.

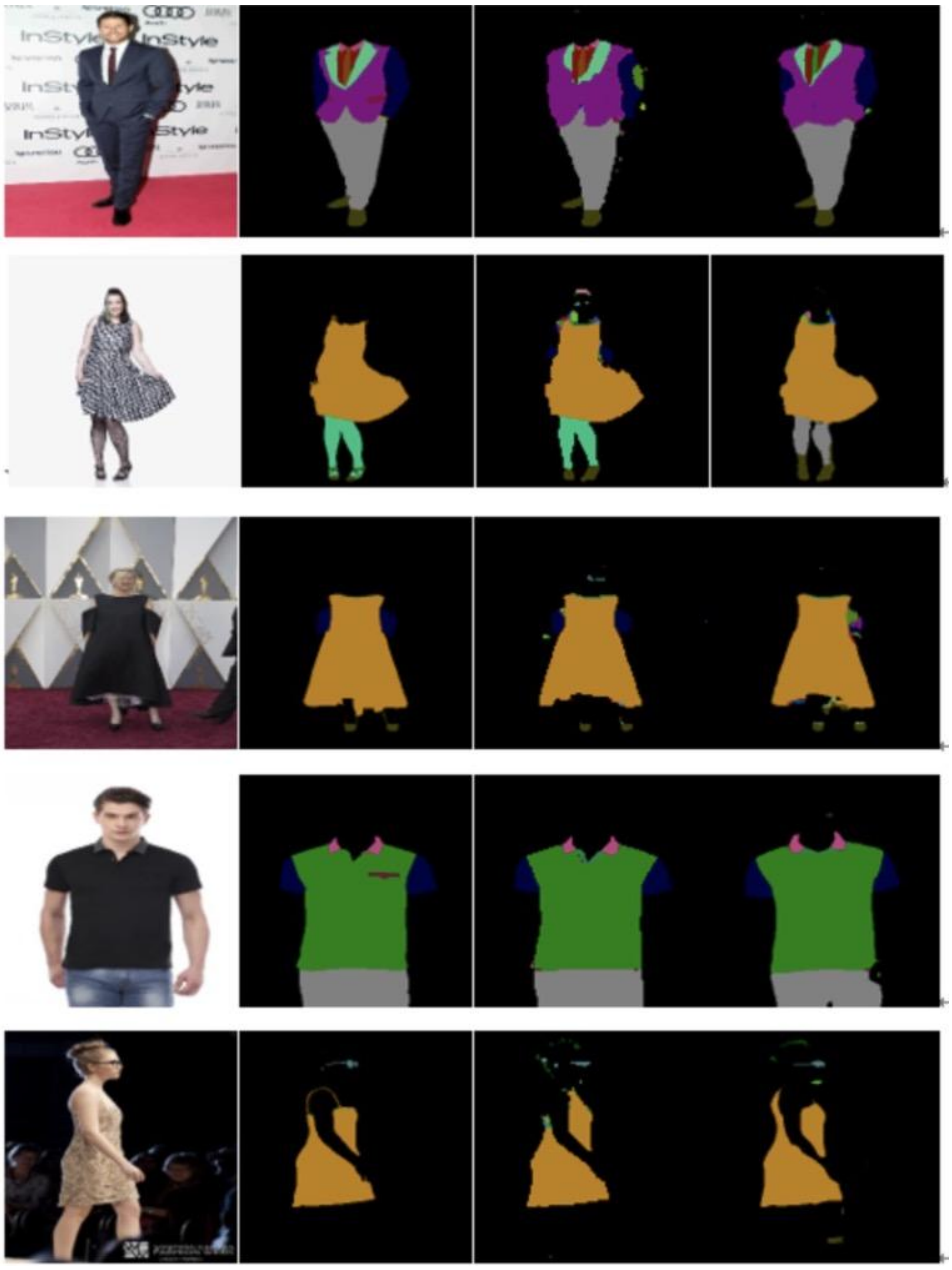
4. Some worse results are caused by subpart-level classification error (Pic 5). It shows the drawbacks of semantic segmentation methods in clothing segmentation tasks: these methods just focus on a small region of image, thus failing to grasp the global feature of the object.

Pic 5. Samples with bad prediction results.

This drawback shows that a top-bottom method may perform well in clothing segmentation tasks. A top-bottom method, like instance segmentation, firstly classifies the whole cloth to make sure the whole part is classified into the same class.

5.6 Visualization results

Pic 6 shows some visualization results. The first column shows some input images. While the second column shows the corresponding ground truth. The third column shows Unet prediction results.The last column shows deeplabV3 + prediction results.`



Pic 6 Visualization results

6. Project execution plan

The main work and estimated time of this project is:

- Dealing with the dataset (4h) - Hongtao Yang
- Designing segmentation models (12h) - Hongtao Yang
- Code implementation of different loss function (8h) - Jiayi Song
- Code implementation of training, test and predict (10h) - Jiayi Song
- Integrating the codes and conducting experiments (12h) - Hongtao Yang
- Analyzing the results and drawing conclusion (4h) - Jiayi Song
- Writing final reports (5h) - Jiayi Song

7. Feasibility and limitations of approach (GPUs, CPUs needed, no of experiments)

Due to computation limitations and lots of experiments, we take two approaches: Firstly, we use 5000 samples to train the model, which is about 1/8 of the dataset. Secondly, we use lightweight models. These approaches help us to conduct sufficient experiments to draw the conclusion. In future, experiments on the whole dataset and large models can be conducted with enough computing resources and time.

8. Potential impact

In our project, we use UNet/deeplab V3+ to train a clothing segmentation model and output masks of some common clothing. This has practical application. Our model is trained with exhaustive classes and you can adjust the concrete output classes according to your needs. Besides, by analyzing the results of different experiments, we find the best loss function in clothing semantic segmentation tasks. It benefits the future study of this task. What's more, our work may inspire other semantic segmentation tasks with large appearance variants.

References

[1] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." *International Conference on Medical image computing and computer-assisted intervention*. Springer, Cham, 2015.

[2] Chen, Liang-Chieh, et al. "Encoder-decoder with atrous separable convolution for semantic image segmentation." *Proceedings of the European conference on computer vision (ECCV)*. 2018.

[3] Xie, Enze, et al. "SegFormer: Simple and efficient design for semantic segmentation with transformers." *Advances in Neural Information Processing Systems* 34 (2021): 12077-12090.

[4] Lin, Tsung-Yi, et al. "Focal loss for dense object detection." Proceedings of the IEEE international conference on computer vision. 2017.

[5] Milletari, Fausto, Nassir Navab, and Seyed-Ahmad Ahmadi. "V-net: Fully convolutional neural networks for volumetric medical image segmentation." 2016 fourth international conference on 3D vision (3DV). IEEE, 2016.

[6] Crum, William R., Oscar Camara, and Derek LG Hill. "Generalized overlap measures for evaluation and validation in medical image analysis." *IEEE transactions on medical imaging* 25.11 (2006): 1451-1461.

[7] Martinsson, John, and Olof Mogren. "Semantic segmentation of fashion images using feature pyramid networks." Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops. 2019.

[8] Ji, Wei, et al. "Human-centric clothing segmentation via deformable semantic locality-preserving network." IEEE Transactions on Circuits and Systems for Video Technology 30.12 (2019): 4837-4848.

[9] Guo, Sheng, et al. "The materialist fashion attribute dataset." *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 2019.

[10] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

Appendix

Snapshot of codes

We use tensorflow.keras.backend to write loss functions. Below shows the realization of some losses.

A1. Miou loss

```
def miou_loss(weights=None, num_classes=2):
    if weights is not None:
        assert len(weights) == num_classes
        weights = tf.convert_to_tensor(weights)
    else:
        weights = tf.convert_to_tensor([1.] * num_classes)

    def loss(y_true, y_pred):
        y_pred = backend.softmax(y_pred)

        inter = y_pred * y_true
        inter = backend.sum(inter, axis=[1, 2])

        union = y_pred + y_true - (y_pred * y_true)
        union = backend.sum(union, axis=[1, 2])

        return -backend.mean((weights * inter) / (weights * union + 1e-8))

    return loss

def weight_miou_loss(weights=None, num_classes=2):
    if weights is not None:
        assert len(weights) == num_classes
        weights = tf.convert_to_tensor(weights)
    else:
        w = [1.0]*num_classes
        w[0] = 0.1
        weights = tf.convert_to_tensor(w)
    def loss(y_true, y_pred):
        y_pred = backend.softmax(y_pred)
        inter = y_pred * y_true
        inter = backend.sum(inter, axis=[1, 2])
        union = y_pred + y_true - (y_pred * y_true)
        union = backend.sum(union, axis=[1, 2])
        return -backend.mean((weights * inter) / (weights * union + 1e-8))
    return loss
```

A2. Dice loss

```
def dice_loss(smooth=1e-2):
    def loss(y_true, y_pred):
        y_pred = backend.softmax(y_pred)
        intersection = backend.sum(y_true * y_pred, axis=[1, 2])
        dice_conf = (2 * intersection + smooth) / (backend.sum(y_true, axis=[1, 2]) + backend.sum(y_pred, axis=[1, 2]) + smooth)
        return 1 - backend.mean(dice_conf)
    return loss
```

A3. Focal loss

```
def focal_loss(alpha=0.25, gamma=2.0):
    def loss(y_true, y_pred):
        y_pred = backend.softmax(y_pred)
        # compute ce loss
        cross_entropy = backend.categorical_crossentropy(y_true, y_pred, from_logits=False)
        # compute weights
        weights = backend.sum(alpha * backend.pow(1 - y_pred, gamma) * y_true, axis=-1)
        return backend.mean(backend.sum(weights * cross_entropy, axis=[1, 2]))

    return loss
```

A4. Cross entropy loss

```
def categorical_crossentropy_with_logits(y_true, y_pred):
    # compute cross entropy
    y_pred = backend.softmax(y_pred)
    cross_entropy = backend.categorical_crossentropy(y_true, y_pred, from_logits=False)

    # compute loss
    loss = backend.mean(backend.sum(cross_entropy, axis=[1, 2]))
    return loss
```

A5. 17- classes dictionary

```
1   background
2   shirt, blouse
3   top, t-shirt, sweatshirt
4   sweater
5   jacket
6   pants
7   shorts
8   skirt
9   dress
10  glasses
11  tie
12  tights, stockings
13  shoe
14  collar
15  lapel
16  sleeve
17  neckline
```

A6. 48 - classes dictionary

```
background
shirt, blouse
top, t-shirt, sweatshirt
sweater
cardigan
jacket
vest
pants
shorts
skirt
coat
dress
jumpsuit
cape
glasses
hat
headband, head covering, hair accessory
tie
glove
watch
belt
leg warmer
tights, stockings
sock
shoe
bag, wallet
scarf
umbrella
hood
collar
lapel
epaulette
sleeve
pocket
neckline
buckle
zipper
applique
bead
bow
flower
fringe
ribbon
rivet
ruffle
sequin
tassel
```

、

A7. ce_train_loss result (results for other loss function all in the github “log_file_result” folder)

	Unnamed:	Wall time	Step	Value
0	0	1.67E+09	0	0.056245
1	1	1.67E+09	1	0.043992
2	2	1.67E+09	2	0.040027
3	3	1.67E+09	3	0.036687
4	4	1.67E+09	4	0.034422
5	5	1.67E+09	5	0.032409
6	6	1.67E+09	6	0.030622
7	7	1.67E+09	7	0.029079
8	8	1.67E+09	8	0.028251
9	9	1.67E+09	9	0.027062
10	10	1.67E+09	10	0.026048
11	11	1.67E+09	11	0.02437
12	12	1.67E+09	12	0.023812
13	13	1.67E+09	13	0.022779
14	14	1.67E+09	14	0.021984
15	15	1.67E+09	15	0.021328
16	16	1.67E+09	16	0.02059
17	17	1.67E+09	17	0.019783
18	18	1.67E+09	18	0.018891
19	19	1.67E+09	19	0.017866
20	20	1.67E+09	20	0.0176
21	21	1.67E+09	21	0.016805
22	22	1.67E+09	22	0.016031
23	23	1.67E+09	23	0.014933
24	24	1.67E+09	24	0.014148
25	25	1.67E+09	25	0.013998
26	26	1.67E+09	26	0.013404
27	27	1.67E+09	27	0.012812
28	28	1.67E+09	28	0.012452
29	29	1.67E+09	29	0.01263
30	30	1.67E+09	30	0.011011
31	31	1.67E+09	31	0.010333
32	32	1.67E+09	32	0.009835
33	33	1.67E+09	33	0.009262
34	34	1.67E+09	34	0.008626
35	35	1.67E+09	35	0.009605
36	36	1.67E+09	36	0.008927
37	37	1.67E+09	37	0.007886
38	38	1.67E+09	38	0.006975
39	39	1.67E+09	39	0.006912
40	40	1.67E+09	40	0.006234
41	41	1.67E+09	41	0.006891
42	42	1.67E+09	42	0.006317
43	43	1.67E+09	43	0.00579
44	44	1.67E+09	44	0.00523
45	45	1.67E+09	45	0.004924
46	46	1.67E+09	46	0.004752
47	47	1.67E+09	47	0.004617
48	48	1.67E+09	48	0.004461
49	49	1.67E+09	49	0.005748
50	50	1.67E+09	50	0.0059
51	51	1.67E+09	51	0.004187
52	52	1.67E+09	52	0.003842
53	53	1.67E+09	53	0.003658
54	54	1.67E+09	54	0.003565
55	55	1.67E+09	55	0.003444
56	56	1.67E+09	56	0.003435
57	57	1.67E+09	57	0.003276
58	58	1.67E+09	58	0.003235
59	59	1.67E+09	59	0.003189
60	60	1.67E+09	60	0.003134
61	61	1.67E+09	61	0.003049
62	62	1.67E+09	62	0.002982
63	63	1.67E+09	63	0.002999
64	64	1.67E+09	64	0.00286
65	65	1.67E+09	65	0.002773
66	66	1.67E+09	66	0.002721
67	67	1.67E+09	67	0.002597
68	68	1.67E+09	68	0.002634
69	69	1.67E+09	69	0.002535
70	70	1.67E+09	70	0.002525
71	71	1.67E+09	71	0.002454
72	72	1.67E+09	72	0.002452
73	73	1.67E+09	73	0.002361
74	74	1.67E+09	74	0.002332
75	75	1.67E+09	75	0.002332
76	76	1.67E+09	76	0.002264
77	77	1.67E+09	77	0.002262
78	78	1.67E+09	78	0.002221
79	79	1.67E+09	79	0.002228
80	80	1.67E+09	80	0.002153
81	81	1.67E+09	81	0.002162
82	82	1.67E+09	82	0.002146
83	83	1.67E+09	83	0.002117
84	84	1.67E+09	84	0.002099
85	85	1.67E+09	85	0.002085
86	86	1.67E+09	86	0.002069
87	87	1.67E+09	87	0.002057
88	88	1.67E+09	88	0.002059
89	89	1.67E+09	89	0.00203
90	90	1.67E+09	90	0.002035
91	91	1.67E+09	91	0.002026
92	92	1.67E+09	92	0.002029
93	93	1.67E+09	93	0.002004
94	94	1.67E+09	94	0.00201
95	95	1.67E+09	95	0.002004
96	96	1.67E+09	96	0.002012
97	97	1.67E+09	97	0.002005
98	98	1.67E+09	98	0.001998
99	99	1.67E+09	99	0.001994