

Project 1 Report of CS307 Principle of Database System

Group 11: Li Boyang, Zhang Yule, Shen Jiahao,
Xu Haoxin, Xu Longxiang, Ma Huaiyuan

January 3, 2024

Contents

1	Project Structure	2
1.1	segment model: FreeSDG model	2
1.1.1	FMAug	2
1.1.2	Coupled Segmentation Network	3
1.1.3	Model Stucture in Python File	3
1.2	segment model: U-net	4
1.3	segment model: PSPnet	4
1.4	segment model: HRnet	4
1.5	segment model: Deeplabv3+	5
1.6	predict.exe	5
2	Operation Procedure	6
2.1	predict.exe	6
3	Operation Result	7

1 Project Structure

Our project is combined with a segment model and a predict.exe structure.

1.1 segment model: FreeSDG model

After researching different segment models, we decide to use FreeSDG(Frequency-mixed single-source domain generalization) model for segmentation. The model structure diagram is as below.

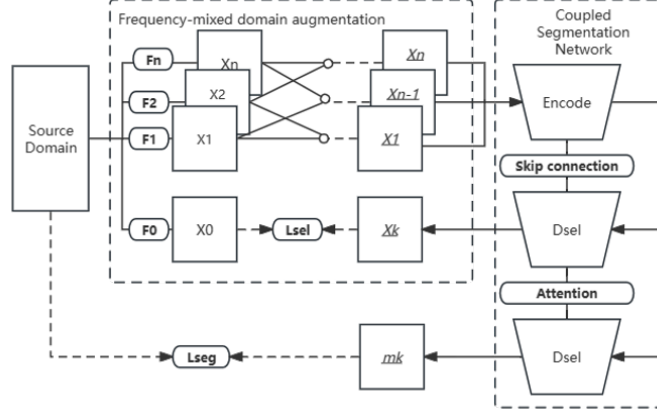


Figure 1: Model structure diagram

As the diagram shows, the model is combined with FMAug(frequency-mixed domain augmentation), coupled Segmentation Network.

1.1.1 FMAug

It is used to identifying the frequency factor for domain discrepancy, extend the margin of the single-source domain. In detailed, it used Guassian-mixup filter.

```
def forward(self, image, mask):
    _, w, h = image.shape

    if self.mixup_size == -1:
        mixup_width = random.randrange(start=0, w)
        mixup_height = random.randrange(start=0, h)
    elif self.mixup_size < -1:
        mixup_width = random.randrange(start=0, -self.mixup_size + 1)
        mixup_height = -self.mixup_size - mixup_width
    else:
        mixup_width = self.mixup_size
        mixup_height = self.mixup_size

    pos_x = random.randrange(start=0, w - mixup_width)
    pos_y = random.randrange(start=0, h - mixup_height)

    hfc_filter_1 = random.choice(self.hfc_list)
    hfc_filter_2 = random.choice(self.hfc_list)

    result_1 = hfc_filter_1(image, mask)
    result_2 = hfc_filter_2(image, mask)

    result_1[:, :, pos_x:pos_x + mixup_width, pos_y:pos_y + mixup_height] = \
        result_2[:, :, pos_x:pos_x + mixup_width, pos_y:pos_y + mixup_height]
```

Figure 2: Gaussian-mixup Diagram

1.1.2 Coupled Segmentation Network

- Optimiazer: Adim
- mechanism: early stop mechanism
- encoder and decoder: U-net architecture with 8 layers
- quantitative index: DICE/Mcc

1.1.3 Model Stucture in Python File

- *init*: Initializes the model by seting various parameters of the model, including network architecture, loss functions, optimizer, etc. Creates the generator network netG, defines loss functions, and initializes the optimizer. Determines visualization variables, model names, etc., based on whether it's in training or testing mode.
- *set-input*: Sets the input data. Depending on whether it's in training mode, determines the input data to be used, including original images, masks, segmentation labels, etc. Applies preprocessing to the images.
- *forward*: Defines the forward pass. Generates the model outputs, including generated images and segmentation results.
- *test*: Runs the model during the testing phase. Computes the confusion matrix, updates evaluation metrics, and calculates visualization variables.
- *train*: Runs the model during the training phase. Sets the model to training mode to ensure learnable parameters can be updated.

```
class FreesdgSegmentationModel(BaseModel):
    @staticmethod
    def modify_commandline_options(parser, is_train=True):...

    def __init__(self, opt):...

    def set_input(self, input, isTrain=None):...

    def forward(self):...

    2 usages (1 dynamic)
    def load_networks(self,.pth_prefix, do_print=True):...

    2 usages
    def __patch_instance_norm_state_dict(self, state_dict, module, keys, i=0):...

    4 usages (3 dynamic)
    def compute_visuals(self):...

    def test(self):...

    def train(self):...

    1 usage
    def backward_G(self):...

    3 usages (3 dynamic)
    def optimize_parameters(self):...

    6 usages (6 dynamic)
    def get_metric_results(self):...
```

Figure 3: model file structure diagram

1.2 segment model: U-net

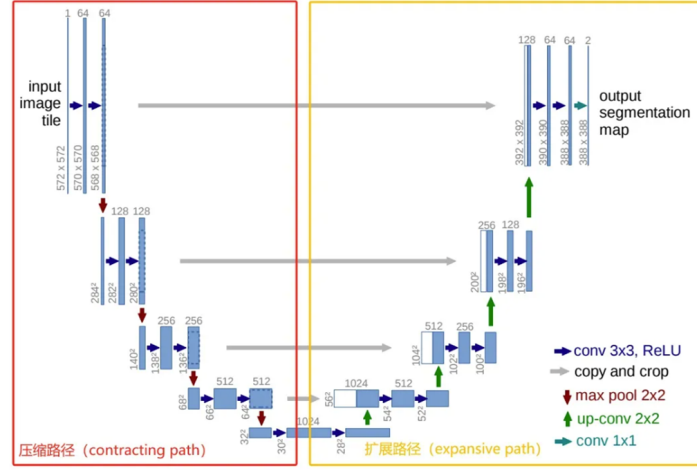


Figure 4: Model structure diagram

Here are three important parts. The input image is an image with edges mirrored. The left part here is compression path with 4 blocks, each block uses two convolutional layers followed by one max pooling layer. The right part is expansion path with 4 blocks, each having one up-sampling deconvolution layer followed by two convolutional layers.

As we know, the compression path is used to capture content information, while the expansion path is used for pixel localization. So skip connection is introduced, achieving a one-to-one correspondence between content and pixel positions. They allow for the reuse of feature maps from the encoder, reducing redundant computations in the decoder.

1.3 segment model: PSPnet

In this model, feature Map refers to the feature map obtained through a Convolutional Neural Network (CNN) that is a pre-trained ResNet. While the most important part is pyramid pooling module. In the diagram, the box labeled **pool** represents the operation of obtaining feature maps of multiple sizes through pooling operations with different sizes of 1x1, 2x2, 3x3, and 6x6. These feature maps are then passed through a "1x1 Conv" to reduce the number of channels; Bilinear interpolation is used for up-sampling to obtain feature maps of the same size as before the pyramid module. These are then concatenated along the channel dimension.

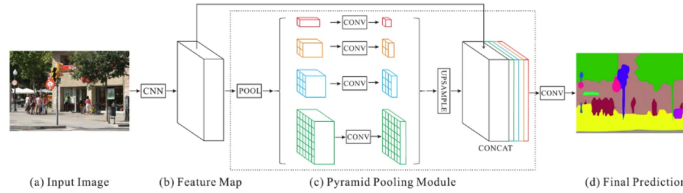


Figure 5: Model structure diagram

1.4 segment model: HRnet

Since the HRnet model is complicated, we divide it into two part: basic part and transition part. In basic structure, stage 1 use a BottleNeck, which is a residual block composed of two convolutional layers. While in stage 2, 3 and 4, basic block is a bottleneck block, utilizing three convolutional layers.

In transition structure, stage 1 is a transitionLayer, this is used to generate a feature map with twice the branches. Stage 4 is a fuseLayer which is used for the interaction of information between different branches. While stage 2 and 3 with overlay of fuseLayer and transitionLayer.

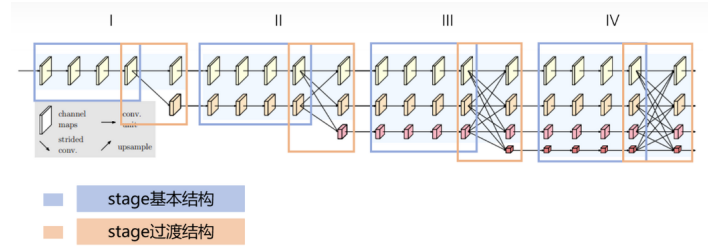


Figure 6: Model structure diagram

1.5 segment model: Deeplabv3+

This model is composed of encoder and decoder. In encoder part, unlike U-net model, serial dilated convolutions are used for feature extraction within the main DCNN backbone; After passing through the main DCNN network, the output is divided into two parts, one part is directly fed into the Decoder, the other part goes through parallel dilated convolution layers for feature extraction, then the features are merged and finally compressed by a convolution layer.

In decoder part, the input is of two part. One part is the output of the DCNN, and the other part is the result of the DCNN output after parallel dilated convolutions. These two results are processed and then combined.

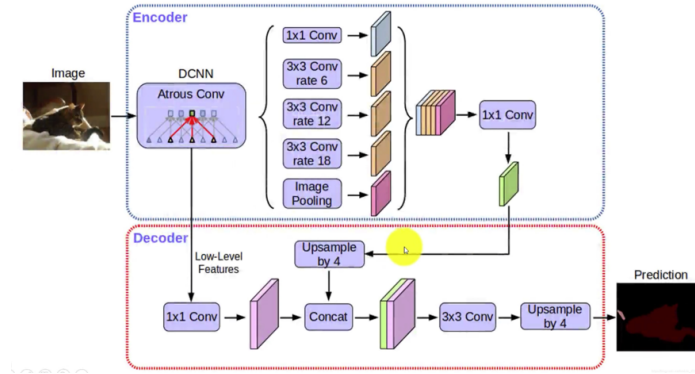


Figure 7: Model structure diagram

1.6 predict.exe

The main procedure is that, this executable file will implement the required environment and initialize the model. It get the input images into the 'captureImage' folder, run predict.py and put the segment result into the 'segout' folder.

```

├── captureImage
├── Model
├── predict.exe
├── models
├── predict.py
├── seg.cpp
└── segout

```

The planned work flow of this navigation platform is shown as below.

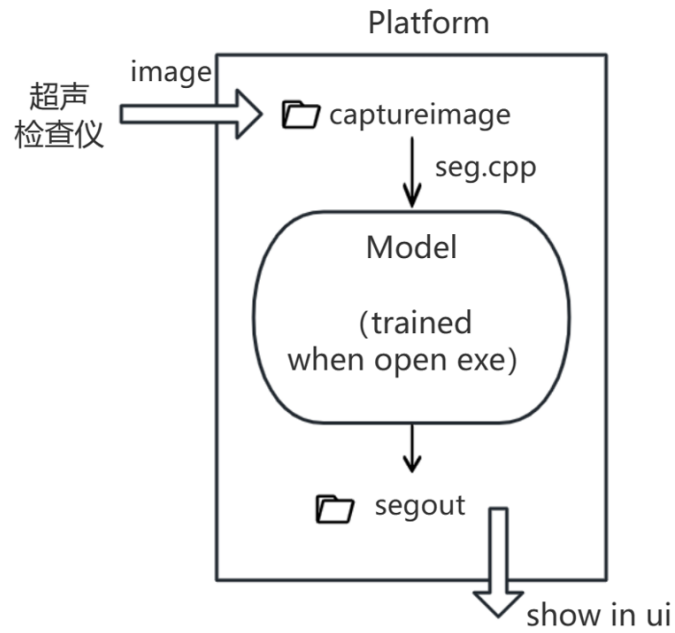


Figure 8: work flow

2 Operation Procedure

After executing the predict.exe, the interface is showed as below.

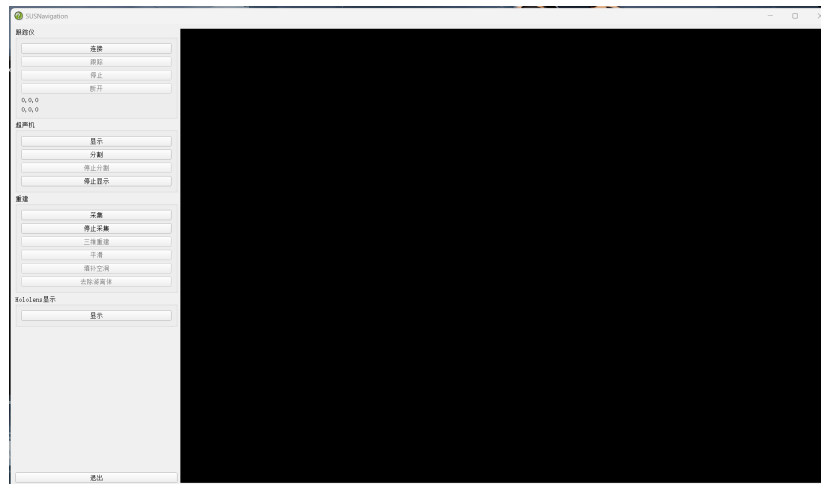


Figure 9: Interface diagram

2.1 predict.exe

1. Connect: the software will connect with the tracker when the button is clicked.
2. Capture and Stop Capture: after connecting, gathering pictures from the tracker by frames.
3. Segment: after gathering pictures, the segment model is applied to the images and show the result when this button is clicked.

3 Operation Result

The result of primary experiment with chosen model is as below. The value in the image refers to MIoU.





























































超声 图像											\
数据 标签											\
DeepL abv3+											54.46
HRNet											88.15
PSPnet											80.55
U-Net											84.58

Figure 10: Interface diagram

Since the facility connected to SUSNaviagation is still under test, we don't have the operation result of the SUSNavigation.