Algorithm Report

In this model optimization process, we made various improvements to enhance the performance of the model. The following will provide a detailed explanation of the optimization methods used to significantly improve the metrics and their improvements compared to the original model.

1. Model and Dataset Overview

The model we adopted and optimized is the UAED (Unet++ and EfficientNet as the encoder). All subsequent optimization efforts were carried out on this model.

The dataset we used is the ‘Narrabeen’. The images are stored in a directory and are read through a normalized CSV file. The CSV file contains the following columns:

{path,rectified,site,camera,type,obstruction,downward,low,shadow,label}

The main columns are ‘path’ and ‘label’, which represent the storage path of the image and the coordinates of the coastline, respectively. Other columns only include category information of images.

This structure ensures a standardized way of accessing and processing the dataset, facilitating consistent data handling during training and evaluation phases.

2. Introduction to Optimization Methods

2.1 Learning rate scheduling strategy

In the initial model, the learning rate decreases from a fixed rate, and the learning rate of each epoch is 0.1 times that of the previous epoch. Compared with the fixed rate reduction scheduling strategy, we adopted a learning rate scheduling strategy of "warmup+cosine annealing".

2.1.1 Warmup

The default parameter for warmup is 5, and during the first 5 epochs of training, the learning rate increases linearly, from 1/5 of the initial learning rate init_1r to init_1r. The specific formula is:

$$lr = init\_lr \times (\frac{epoch+1}{warmup\_epochs})$$

You can adjust the warmup epochs while training by add the argument: (also need to adjust the max epoch)

```
python train_uaed.py --warmup=5 --maxepoch=25
```

2.1.2 Cosine annealing

In the epochs after the warmup stage, the learning rate gradually decreases according to cosine annealing, with the specific formula being:

$$lr = 0.5 \times init\_lr \times (1 + cos(\pi \times \frac{epoch - warmup\_epochs}{total\_epochs - warmup\_epochs}))$$

Cosine annealing can cause the learning rate to exhibit periodic decay during the training process, which helps to escape from local optima and find a better global optimum.

2.2 Positive sample expand

In order to enhance the recognition ability of the model for the target object, we performed a "thickness" process on the positive samples, dilated the labels to make them thicker, and then added these thickness labels to the original label image. This strategy can enhance the model's ability to recognize the shorelines.

2.3 Channel number adjustment

In the original model, the number of channels for the decoder is set to (256, 128, 64, 32, 16). In order to reduce the number of model parameters and improve computational efficiency, we adjusted the number of channels to (128, 64, 32, 16, 8). Reducing the number of channels helps to lower model complexity and computational overhead, reduce the risk of overfitting, and maintain high feature extraction capabilities through reasonable network structure design.

2.4 Convolutional Kernel Size Adjustment

The convolution kernel size used in the original model was 3x3, but we adjusted it to 5x5. A larger convolution kernel can increase the receptive field, allowing the model to capture a larger range of spatial information, thereby enhancing the ability of feature extraction.

2.5 Data augmentation

A series of data augmentation strategies have been adopted to improve the generalization ability of the model. These enhancement strategies include:

- Randomly adjust the contrast of the image between 0.5 and 1.5.
- Randomly adjust the saturation of the image between 0.5 and 1.5.
- Randomly adjust the brightness of the image between 0.5 and 1.5.
- Randomly rotate the image, with a rotation angle between -15 degrees and 15 degrees.
- Randomly shift the image. The translation amount randomly varies within ± 10% of the target size.
- Randomly scale the image with a scaling ratio ranging from 0.8 to 1.4.

Due to the relatively single scenario, data augmentation cannot produce a significant improvement in metrics. So during training, data augmentation is set to off by default. However, you can enable the data augmentation while training by add the argument:

```
python train_uaed.py --aug_open=True
```

Over all, the command we used to train is

```
python train_uaed.py --batch_size 8 --csv_path 'train_set.csv' --tmp
'Gsave_path/trainval' --warmup 5 --maxepoch 25
```

In addition to the above methods, we have also implemented other optimization methods, which were not included in the final strategy due to the unclear optimization of model performance, including Leaky-Relu replacement Relu, Modifying the size of the input image, adjusting the weights of positive and negative sample loss, using ema strategy, etc.

3 Optimization Results

3.1 Post processing

The coastline prediction image directly output by the model is a grayscale image representing a probability distribution, and the result is usually coarse. To obtain the coastline, a series of post-processing steps are required.

- Set a threshold of 200 to convert the probability distribution grayscale image into a binary image.The specific method is to set the parts with pixel values greater than or equal to 200 to 1, and the rest to 0.
- Use morphological closing operations to make the prediction image smoother, fill small holes, and connect adjacent areas.
- Perform a skeleton extraction algorithm to make the shoreline one pixel wide.
- Remove small noise points and artifacts in the image.

Through these post-processing steps, the final predicted coastline is a one-pixel wide line with high accuracy and clarity.

3.2 Metric introduction

ODS is a commonly used metric in edge detection, and we have modified it based on it to make it more suitable for our project. It calculates precision, recall, and F1 score by comparing the binary prediction graph (pred-binary) with the binary truth graph (gt-binary) at a given distance threshold. We traverse each predicted point, and if its distance to the nearest truth point is less than the threshold and the truth point is not matched, we count the number of successful points and mark that truth point has been matched. The final F1 score calculated will serve as the final metric.

3.3 Results

We conducted tests on the ‘Narrabeen’ scenario. The following table shows the metrics of the default model and the metrics after each optimization method. Each metric is obtained by adding a new optimization method based on the previous model.

Models/Optimization method	ODS
Default model	0.8237039661717135
Cosine annealing	0.82543689914072
Warmup	0.8288359617600757
Pos Expand	0.8350900982119961
Channel adjust	0.8352136918340278
Kernel adjust (final model)	0.8387546893779404

3.4 Resource consumption

Our model training and inference mainly rely on GPU computing, using NVIDIA T4 GPU. The total average time per iteration for the default model and the optimized model are 2.78 seconds and 2.96 seconds, respectively. While the model metrics increase, resource consumption is effectively controlled by balancing the number of channels and convolution kernel size.