

Research on Semantic Segmentation Method of Urban Street Scene Based on DeepLabv3+

Huazhi Lin

2200376

(of Affiliation)

Software Engineering and Artificial

Intelligence program

2200376@student.uwtsd.ac.uk

I. INTRODUCTION

In 1943, the first article on artificial neural networks was published by a neurologist W.S. McCulloch and a mathematician W. Pitts, introducing the first mathematical model of an artificial neuron - the MCP model. Since then, the development of artificial neural networks has gradually shifted from simple to complex and from linear to nonlinear models, and has been accompanied by rapid improvements in computer hardware, particularly in GPU performance, leading to a flourishing period for neural networks.

In recent years, with the rapid increase in the number of cameras, neural networks have been widely used in intelligent video quality analysis [1-5]. Scene understanding is typically required to determine whether the monitoring setup is reasonable. Semantic segmentation is one of the key technologies in computer vision [6], and its implementation is a necessary prerequisite for achieving scene understanding. In urban street scenes, semantic segmentation technology is often used to segment objects of interest, such as pedestrians, vehicles, roads, traffic signs, and road dividers. However, due to the complexity of urban street scenes and the presence of many moving targets, occlusions between objects, and the impact of weather and lighting, segmentation tasks are extremely challenging. In recent years, many researchers have provided benchmark datasets by finely annotating captured urban road scene images, such as Cityscapes [7], ApolloScape [8], among others. A portion of the annotated information in Cityscapes dataset is shown in Figure 1.1.

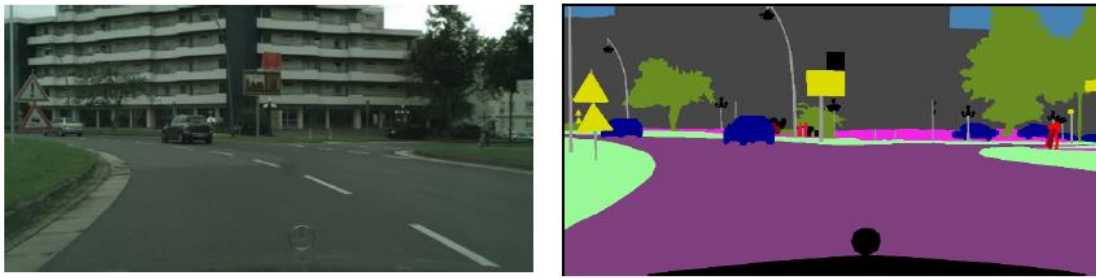


Fig. 1.1. Cityspace dataset partial annotation.

Researchers have carried out fine pixel-level annotations on objects of different categories, including pedestrians, vehicles, vegetation, buildings, roads, bridges, traffic signs, signal lights, and many other classes. This has provided researchers with a large-scale dataset that includes different cities, weather and lighting conditions, which can be used for semantic segmentation research aimed at urban street scenes. Semantic segmentation technology classifies each pixel point in the scene image and distinguishes objects by pixel color. Unlike object detection, this technology can accurately predict the boundaries of objects, providing precise and critical information source for autonomous driving with important practical applications.

II. LITERATURE REVIEW

A. Traditional semantic segmentation methods

Traditional semantic segmentation methods typically utilize shallow features of the image, such as grayscale, texture, and geometry, to divide the image into multiple non-overlapping regions. The objective is to ensure that these features are consistent or similar within the same region but different between different regions. Several common traditional methods are described below:

(1) Threshold-based semantic segmentation method

This method can be split into two types: single threshold segmentation and adaptive threshold segmentation [9]. The former employs a fixed threshold to segment the image; for example, the classic mode method based on the gray-scale image's bimodal histogram is a common global single-threshold segmentation method. This method assumes that there is a distinct difference between the background and the target in the image, resulting in a bimodal distribution of the gray-level histogram. Selecting the valley between the two peaks as the segmentation threshold yields good results for separating the target from the background.

Conversely, the latter calculates the threshold applicable to each region according to the brightness distribution of the different regions in the image. Therefore, the adaptive threshold segmentation method is also called a local threshold segmentation method. Classic adaptive threshold segmentation methods include Otsu's one-dimensional maximum interclass variance algorithm [10], Liu Jianzhuang's two-dimensional maximum interclass variance method [11], which improves the noise immunity performance, and Jing Xiaojun's three-dimensional interclass variance method [12], which achieves good results even under low contrast and signal to noise ratio conditions. However, the threshold-based semantic segmentation method only performs statistical analysis of the pixel points, lacks overall information capture, ignores the spatial relationships of the entire image, and has low segmentation precision because it is highly influenced by noise interference.

(2) Region growing segmentation method

The region growing segmentation method was originally proposed by Levine et al. [13], and its basic idea is to group pixel sets with similar characteristics to form regions. This method typically starts with several seed points inside the target object. From these points, adjacent and similar pixel points or other regions are merged into the current region to achieve seed region growth. The similarity measurement between regions includes image information such as average grayness, texture, and geometry. Although the region growing segmentation method can often achieve good performance without prior knowledge, it is an iterative algorithm with high spatial and temporal overhead. Moreover, many factors, such as seed region selection and termination conditions, affect the region growing method; therefore, its applicable scenarios are limited. To address problems where region growing segmentation performs poorly under complex conditions, other segmentation methods, such as edge detection and threshold segmentation, can be used.

(3) Watershed segmentation method

The watershed segmentation method [14] regards the grayscale image of an image as a three-dimensional map in geometry, with each pixel's gray value as its height. Local minima and their influence areas are regarded as basins, and the boundaries between the basins are seen as watersheds. This method is an iterative calculation process. Vincent proposed a watershed segmentation method that includes two steps: sorting and flooding. First, sorting the image pixel gray values, and then judging whether the point is flooded based on the gradient function, implementing the flooding process from low to high, and finally obtaining multiple non-overlapping pixel regions and continuous region boundaries. However, because it responds well to weak edges, this method tends to over-segment the image due to noise or weak gray transitions on the target surface. To address this issue, prior knowledge can be used to remove irrelevant edge information or modify the gradient function so that basin expansion only responds to the target that needs to be detected, resulting in better image segmentation results.

(4) Graph cut algorithm

The graph cut algorithm [15] is an energy optimization algorithm that associates the image segmentation problem with the minimum cut problem of the graph. This algorithm uses an undirected graph $G=(V, E)$ to represent the image to be segmented, where V and E represent vertices and edges in the image. Unlike ordinary graphs, the graph cut algorithm also adds two extra terminal vertices S and T , connecting all vertices with these two vertices to form part of the graph. Unlike ordinary graphs, edges not only have direction but also carry weights. By using the minimum cut algorithm, the smallest edge connecting S and T is found to achieve image segmentation.

B. A Semantic Segmentation Method Based on Convolutional Neural Networks

The emergence of the MCP model has facilitated the rise and evolution of artificial neural networks, with Convolutional Neural Networks (CNN) emerging as a crucial approach in deep learning, exhibiting strong feature extraction capabilities. Traditional semantic segmentation methods focus on pixel-level computation and prediction, leading to loss of spatial information. However, CNN can enlarge the receptive field simultaneously during feature extraction, enabling the extraction of global image features. As a result, CNN has gradually become the preferred method for semantic segmentation. Several common network architectures based on CNN for semantic segmentation are presented below.

(1) Fully Convolution Networks, FCN

The Fully Convolutional Networks (FCN) model proposed by Jonathan Long et al. utilises the convolution layer weights of existing networks, such as AlexNet and VGG16, to construct a new architecture (as shown in Figure 2.1), accomplishing training for segmentation tasks through fine-tuning. In 2014, FCN-8s achieved a mean IoU of 62.7% on the PASCAL VOC dataset, improving performance by about 20%. FCN network extracts features that are the result of feature fusion from different convolution layers in the encoder, containing both high-level and low-level features with preserved spatial information. Moreover, FCN utilises Transposed Convolution to upsample low-resolution features, and initialises the weights of Transposed Convolution using bilinear interpolation. This approach enables the recovery of original image spatial information through upsampling, while avoiding the high computational cost of using dilated convolution.

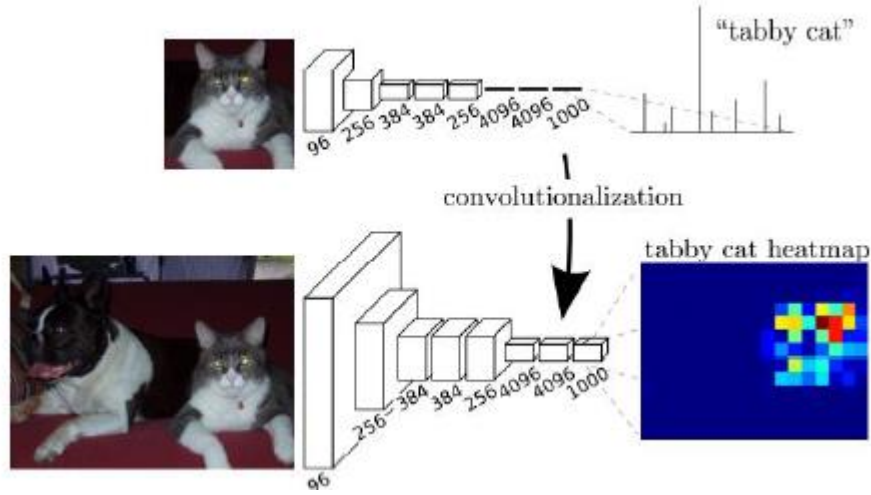


Fig. 2.1. FCN structure diagram.

(2) SegNet

The SegNet [18] is innovative in its approach to upsampling low-resolution input features. Through preserving the max-pooling indices in the encoder's max-pooling layer, SegNet is able to perform upsampling. This method means that the decoder used for upsampling no longer needs to be trained because the sparse feature map obtained by upsampling using the max-pooling indices retains high-frequency information. Dense feature maps can then be generated through convolution, which is a learnable operation. Results show that when compared with widely used networks such as FCN and DeconvNet [19], SegNet achieves a balance between memory consumption and segmentation accuracy. The process of unpooling in SegNet is illustrated in Figure 2.2.

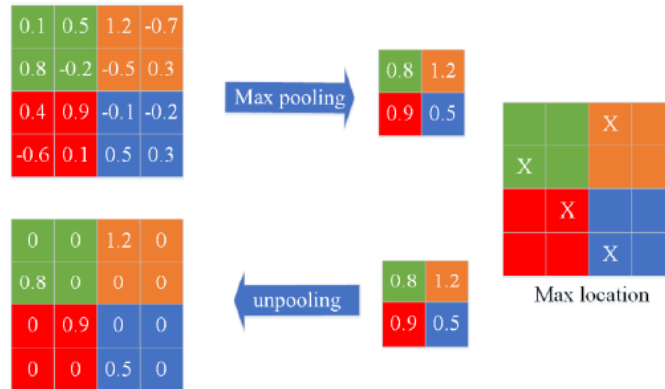


Fig. 2.2. SegNet's anti pooling process.

C. U-Net

The main contribution of U-Net [20] is its symmetric U-shaped structure (as shown in Figure 2.3), also known as an encoder-decoder architecture. The first half of the network is used for feature extraction, while the second half involves upsampling the deep features extracted and restoring the spatial information lost during the extraction process. U-Net is a fully convolutional network consisting of convolutional and max-pooling layers in the feature extraction network, while in the upsampling network, the upsampling output feature map of a particular node is concatenated with the corresponding input. Unlike FCN, which adds feature maps in corresponding positions during feature fusion, U-Net uses concatenation to stack feature maps together in the channel dimension and form thicker feature maps. Furthermore, due to U-Net's multilevel upsampling, which combines upsampled output from the previous level (large-scale features) and output from feature extraction at the same level (same-scale features), multi-scale feature fusion pervades the entire network, whereas FCN only performs fusion in the last layer. This architecture is less dependent on the number of images, and even with fewer images for end-to-end training, segmentation accuracy is not significantly reduced.

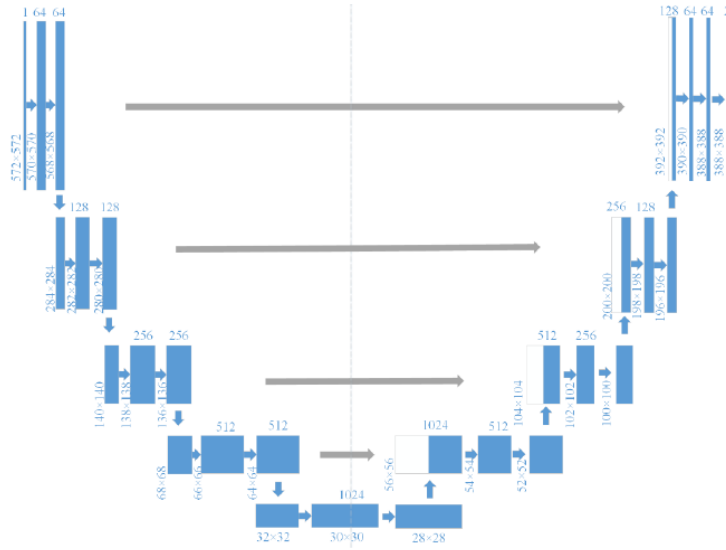


Fig. 2.3. U-Net Network U-shaped Structure.

D. Deeplab series (v1, v2, v3, v3+)

DeepLab is a technique that combines deep convolutional neural networks (DCNNs) and conditional random fields (CRFs) for semantic segmentation. One issue with DCNNs in semantic segmentation tasks is their high-level feature's translation invariance, which persists even after frequent pooling and downsampling, resulting in the loss of location information for fine details.

To address this problem, DeepLabv1 [21] modified VGG16 by replacing its fully connected layers with convolutional layers that had fewer parameters, removing the last two max-pooling layers, and using dilated convolutions in subsequent convolutional layers to increase the receptive field while preventing information loss during downsampling. However, the feature extraction ability of dilated convolutions on fine details declines rapidly as the dilation rate increases. To address this, DeepLabv1 added a fully connected CRF to improve the model's ability to capture details. Although the author also used multi-scale feature fusion inspired by U-Net for prediction, the actual effect was not ideal compared to CRF. The model achieved a miou of 71.6% on the PASCAL VOC 2012 test set.

DeepLabv2 [11] is an improved version of v1 that uses ResNet instead of VGG16 as the backbone network to enhance the feature extraction ability. ResNet solves the problem of degradation in deep networks and enables it to extract deeper semantic information than VGG16. Additionally, ASPP replaces multi-scale prediction and significantly increases the receptive field. ASPP fuses the output of multiple dilated convolutional layers with different dilation rates to obtain the ability to acquire multi-scale features like FPN. The model achieved a miou of 79.7% on the PASCAL VOC 2012 test set. The details of ASPP will be explained in Chapter 3 of this paper.

DeepLabv3 [23] is an improvement over v2. First, the ResNet backbone network was deepened by adding three residual blocks after block4. Additionally, instead of downsampling in block4, new convolutional layers were created using dilated convolutions, reducing the number of sampling times from five to four. Second, the ASPP module was enhanced by using three 3×3 dilated convolutions with dilation rates of 6, 12, and 18 for multi-scale feature extraction. Each convolutional layer contained 256 convolutional kernels and BN layers. The ASPP module also included global average pooling and 1×1 convolutional features extracted from the original image, and these five features were concatenated and refined by a 1×1 convolutional layer as the output. The authors conducted comparative experiments in the paper, using different output_stride values while using ResNet-50 and seven blocks. The results showed that when the output_stride value was larger, the model had a lower miou on the test set, proving the necessity of retaining spatial size. Finally, the model achieved a miou of 85.7% on the PASCAL VOC2012 test set and 86.9% with the JFT dataset.

DeepLabv3+ [24] introduced an Encoder-Decoder structure based on DeepLabv3, using a decoder to upsample deep features and recover spatial information lost during downsampling. The input to the decoder was designed as a multi-scale fusion feature refined by 1×1 convolutional layers after the ASPP structure. The fusion of high-dimensional and low-dimensional features improved segmentation accuracy. The authors also improved the Xception network, which had slightly better performance than the ResNet model when used as the backbone network. In the Xception model, convolution was decomposed into depthwise convolutions and pointwise convolutions, greatly reducing the number of parameters and computation. The DeepLabv3+ model achieved a miou of 87.8% on the PASCAL VOC2012 test set and reached 89% with the JFT dataset.

III. METHODS

We made some improvements to the DeepLabv3+ model to achieve better performance on the Cityscapes dataset.

A. Deeplabv3+network model

DeepLabV3+ is a state-of-the-art semantic segmentation model that incorporates encoder-decoder architecture to effectively fuse multi-scale information. The encoder in DeepLabV3+ extracts features from the input image using a deep convolutional neural network with serial atrous convolutions. The output of the encoder is divided into two parts: low-level features directly passed to the decoder, and parallel atrous convolutions with different rates used for feature extraction to capture contextual information at different scales. The resulting features are merged, compressed using 1x1 convolutions, and sent to the decoder. In the decoder, high-level features are upsampled by a factor of four using bilinear interpolation and concatenated with low-level features that have been compressed using 1x1 convolutions. Then, two depthwise separable convolution blocks are applied to extract features, followed by upsampling to produce a prediction map of the same size as the input image. The model structure is illustrated in Figure 3.1.

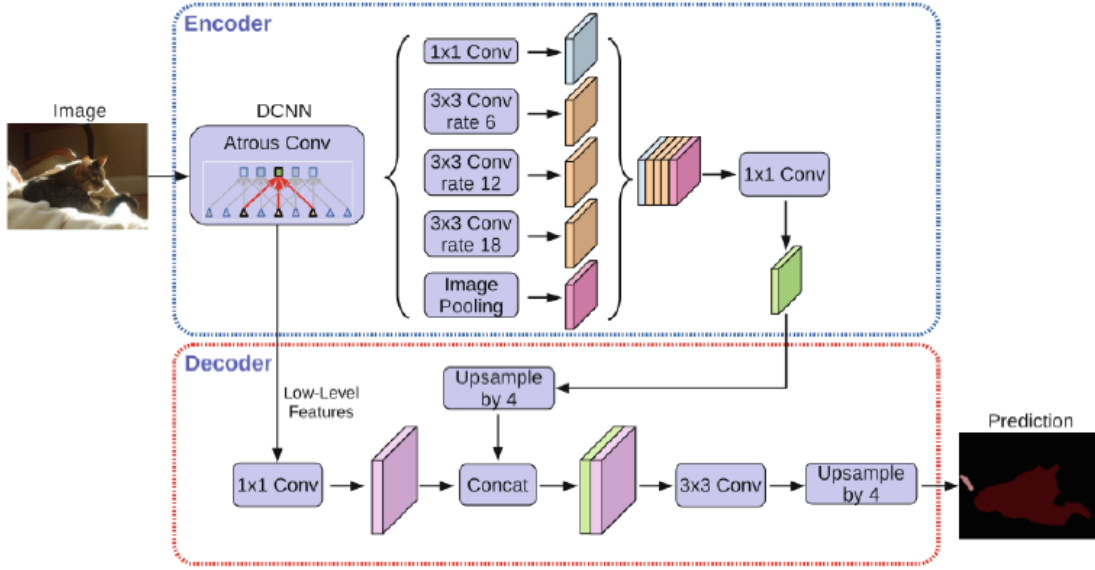


Fig. 3.1. DeepLabv3+network structure diagram

B. Deeplabv3+network model

Although DeepLabv3+ achieves good performance in semantic segmentation of CityScapes street scenes, the limited number of training samples and difficulty of neural network learning different types of objects in complex street scenes lead to uneven distribution in the number of classifications. Therefore, even though DeepLabv3+'s loss value is low, some classifications' accuracy in testing evaluations is often low. To address this issue, the authors use Focal Loss Function to replace Cross Entropy Loss Function to balance the unevenness of each classification in CityScapes dataset [25]. In the original DeepLabv3+ model, the authors used the cross-entropy of the predicted results and ground truths as the pixel loss without adjusting for imbalanced categories in the dataset. During training with Cityscapes dataset, the frequency of road and building classifications in the training set was higher, resulting in an IoU of over 90%. However, for classifications such as bus and truck, the predicted IoU was often less than 65%. This paper provides statistical analysis of the pixel area of each classification in the training set (Table 3.1).

Even with randomly generated training queues using tensorflow, the disparity in the number of targets and pixel quantity among different classifications during a single epoch of training can lead to insufficient feature learning for categories with fewer samples and smaller areas. Thus, label_weights are used as hyperparameters to weight the pixel losses of different labels punishing small samples in the loss function.

This paper attempts to address the issue of class imbalance from two aspects: imbalanced quantity and difficulty. Firstly, label_weights are used to weight the pixel loss to solve the quantity imbalance problem. Secondly, Focal Loss Function is used to replace the Cross Entropy Loss Function to tackle the difficulty imbalance problem during the training process.

Table 3.1. Number of pixels in groundtruth for each classification

classification	Training label	Number of pixels in this classification in Groundtruth (in millions)
pavement	0	238.13

sidewalk	1	38.56
build	2	146.07
railing	3	5.60
vegetation	4	103.74
terrain	5	7.16
sky	6	25.22
pedestrian	7	7.90
Riders	8	0.94
car	9	44.63
goods train	10	1.75
bus	11	1.66

1) Cross entropy loss function

Cross entropy loss function is often used as a loss function for multi category tasks in deep learning. The definition of cross entropy is as follows (3-1):

$$H(P, Q) = - \sum_{i=1}^n P(x_i) \times \log(Q(x_i)) \quad (3-1)$$

Among them: x_i is the i -th sample input; P is the true probability distribution; Q is the predicted probability distribution; $H(P, Q)$ is the cross entropy.

Cross-entropy measures the similarity between two probability distributions, P and Q . When the label is one-hot encoded, for example, the true distribution $P=[0,1,0,0,0]$ and the predicted distribution $Q=[0.2,0.5,0.1,0.1,0.1]$ after softmax, the calculation of cross-entropy only involves one probability value. Although cross-entropy here only uses one probability value, the predicted distribution Q is optimized through softmax [26], implying optimization of the entire probability distribution. The cross-entropy formula in this case can be simplified to equation (3-2):

$$H(P, Q) = -\log(Q(x_i)) \quad (3-2)$$

Therefore, when using cross-entropy as the loss function, the model's logits are typically passed through softmax before calculation to ensure the predicted probabilities for each class are limited within the range of $[0, 1]$. Then, cross-entropy is calculated based on the output vector from softmax and the true one-hot label. The original DeepLabv3+ model also utilizes pixel-wise cross-entropy as the pixel loss, but it fails to address the issue of class imbalance. To tackle it, we improve the cross-entropy loss function by weighting the obtained pixel cross-entropy with hyperparameter `label_weights` to balance the uneven quantity of different classifications. Our `label_weights` hyperparameters are set as $[1., 1.3, 1., 2.3, 1., 1.9, 1.4, 1.8, 10., 1.2, 5., 5.]$, with the weight of the first classification of training ID 0 serving as the baseline, and weights for other classifications being approximately calculated based on their pixel quantities in the training set. In the calculation process, a limit was imposed on the maximum weight to prevent loss from oscillating uncontrollably due to an overly large weight for a certain classification. Equation (3-3) represents the weighted cross-entropy loss function.

$$H(P, Q) = -\alpha \log(Q(x_i)) \quad (3-3)$$

among α is the weighting coefficient.

2) Dichotomous focus loss function

In 2017, KaiMing He et al. addressed the issue that one-stage detectors are less accurate than two-stage detectors and argued that class imbalance is the root cause. Specifically, they noted that the vast majority of candidate boxes generated in object detection do not contain actual objects, resulting in a severe class imbalance problem. Moreover, these negative samples, which occupy a significant proportion of the model's loss function, are typically easy to classify, potentially leading the model to optimize in the wrong direction. To tackle this issue, the authors introduced a new loss function called Focal Loss, which increases the impact of hard examples on the loss function by controlling the weights of easy-to-classify samples. While the standard cross-entropy formula is given by Equation (3-2), the loss formula that balances easy and hard cases should take the form of Equation (3-4).

$$FL(p_t) = -(1 - p_t)^\gamma \log(p_t) \quad (3-4)$$

In order to enable the focus loss function to simultaneously weigh the hard and easy samples and positive and negative samples, the focus loss function draws on formula (3-3) and introduces a weighting coefficient α . The formula is as shown in equations (3-5).

$$FL(p_t) = -\alpha t(1 - p_t)^\gamma \log(p_t) \quad (3-5)$$

3) Multi category focus loss function

Since there are 12 categories of interest in the Cityscapes dataset used in this experiment, the second category focus loss function cannot be applied to the Deeplabv3+ model, so a multi category focus loss function is required. The dichotomous cross entropy loss function and focus loss function are shown in equations (3-2) and (3-5). According to the definition of cross entropy (3-1), equation (3-2) can be written as follows:

$$H(P, Q) = -P(x) \cdot \log(Q(x)) - (1 - P(x)) \cdot \log(1 - Q(x)) \quad (3-6)$$

Expand equations (3-5):

$$FL(p_t) = -\alpha(1 - p_t)^\gamma \cdot y \cdot \log(p_t) - (1 - \alpha) \cdot p_t^\gamma \cdot (1 - y) \cdot \log(1 - p_t) \quad (3-7)$$

If softmax is added to the loss function, since the output of softmax is the probability of each class, and the sum of the probabilities is 1, the given label is always in one hot form when softmax is used, assuming that the output of a binary softmax is p_1, p_2 , and the input labels are y_1, y_2 , then from (3-6) we can get:

$$H(P, Q)_{softmax} = -P(x_1) \cdot \log(Q(x_1)) - P(x_2) \cdot \log(Q(x_2)) \quad (3-8)$$

Add attenuation coefficient to (3-8) γ Obtained:

$$FL(p_t)_{softmax} = -(1 - p_1)^\gamma \cdot y_1 \cdot \log(p_1) - (1 - p_2)^\gamma \cdot y_2 \cdot \log(p_2) \quad (3-9)$$

Add to (3-9) α Coefficient, the first type is α_1 . The second category is α_2 . It can be concluded that:

$$FL(p_t) = -\alpha(1 - p_t)^\gamma \cdot y \cdot \log(p_t) - (1 - \alpha) \cdot p_t^\gamma \cdot (1 - y) \cdot \log(1 - p_t) \quad (3-10)$$

Due to the one hot form of the label, the label of a certain type of sample is only 1 at the subscript of the corresponding class number, and the rest is 0. Therefore, (3-10) can be written as:

$$FL(p_t) = -\alpha(1 - p_t)^\gamma \cdot y \cdot \log(p_t) - (1 - \alpha) \cdot p_t^\gamma \cdot (1 - y) \cdot \log(1 - p_t) \quad (3-11)$$

among α_c is the sample weight of category c , and p_c is the prediction probability of category c . Therefore, the focus loss function of multiple classifications is Formula (3-11), which is also the focus loss function of the second classification.

C. Using ASFF for shallow feature fusion

In the original implementation of the Deeplabv3+ model using the TensorFlow framework, the multi-scale features of the decoder input were directly concatenated along the channel dimension using the concat function without considering the crucial differences between features. Therefore, this paper proposes the use of the Adaptively Spatial Feature Fusion (ASFF) method to adaptively fuse the multi-scale features of the decoder input.

In object detection, Feature Pyramid Networks (FPN) [27] are commonly used to detect objects of varying scales. FPN typically up-samples small-scale features through interpolation, fuses them with large-scale features, and then makes predictions. The specific structure of the FPN and feature fusion process are shown in Figures 3.2(a) and 3.2(b), respectively. The input features of the Deeplabv3+ decoder, on the other hand, are a simple fusion of the features extracted by the encoder and the shallow features of the backbone network, as shown in Figure 3.3, and have a structure similar to that of FPN. Many of the features involved in fusion come from different depths and have different semantic features, so a simple concatenation might result in incomplete feature utilization. To address this issue, this paper proposes the use of ASFF to perform adaptive feature fusion of the shallow features of Xception_65's

'entry_flow/block2/unit_1/xception_module/separable_conv2_pointwise',
'entry_flow/block3/unit_1/xception_module/separable_conv2_pointwise', and
'exit_flow/block1/unit_1/xception_module/separable_conv2_pointwise' in addition to the features extracted by the encoder.

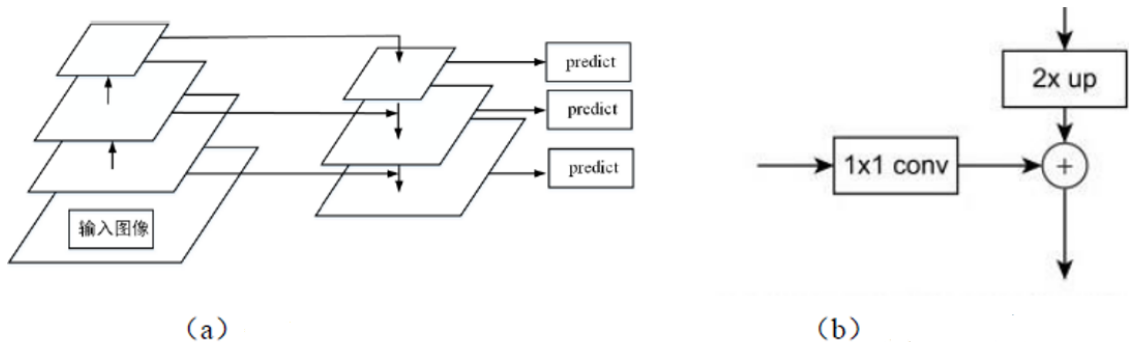


Fig. 3.2. FPN structure

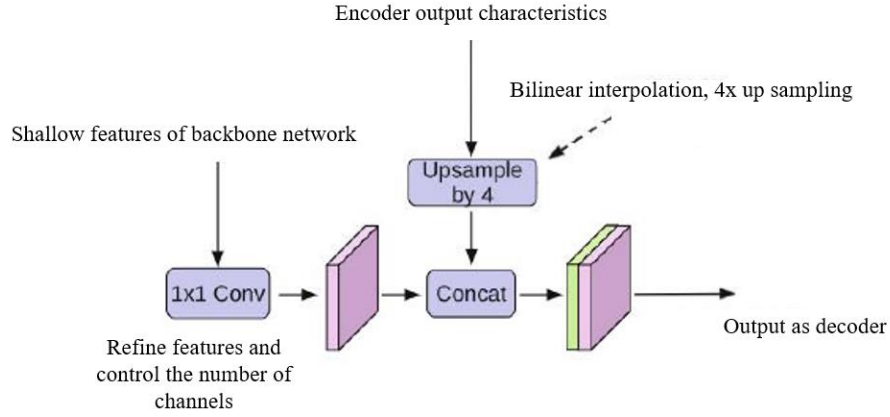


Fig. 3.3. The Fusion Process of Decoder Input in DeepLabv3

1) Improving Decoder with ASFF

The input to the DeepLabv3+ decoder differs from that of FPN in that it does not require multiple branches for multiple predictions, but rather only needs one fused feature.

In the original DeepLabv3+ model, the Decoder input was obtained by concatenating L1/4 features with the output features of the Encoder. The authors found through multiple experiments that the model performed better when the number of shallow layer feature channels was 48. To reuse the Decoder parameters, in this paper, three ASFF-processed features were used and converted to 48 channels via a 1x1 convolution. Specifically, the three shallow features from the backbone network were transformed using a 1x1 convolution to adjust channel numbers, then concatenated along the channel dimension before being refined with a 1x1 convolution and the channel number was adjusted to 3. They were then passed to softmax, weights were adjusted between $[0,1]$, and finally the weight matrix was multiplied with the shallow features along the channel direction to obtain the processed fused feature. The process for calculating the weight matrix is shown in Figure 3.4, and the structural comparison of the model before and after improvement is shown in Figure 3.5.

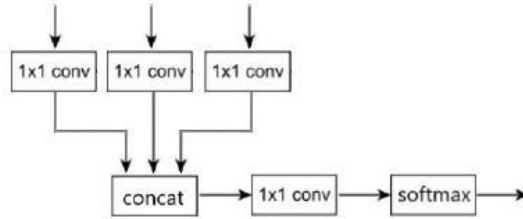


Fig. 3.4. The process of obtaining weights from feature maps

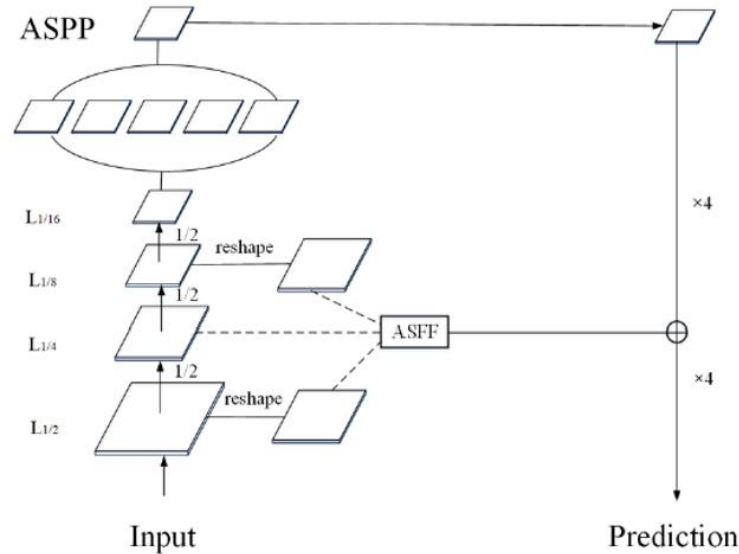


Fig. 3.5. Improved Model Decoder Input Structure

D. Using SENet to Improve Xception

Although Xception has a residual structure similar to ResNet, to make the neural network focus not only on the spatial dimensions of the image like ASPP, but also to select features based on their importance from the feature channel dimension during feature extraction, this paper introduces the SE-block from Squeeze-and-Excitation Networks (SENet) in the Xception backbone network, which serves as an attention mechanism.

1) SENet

SENet was the first to improve network performance by optimizing feature channels rather than just improving spatial dimensions [29]. The authors of SENet proposed optimizing the global loss function of the model to learn the importance of each feature channel and retain important features while suppressing less significant ones. Although this module introduces additional computational complexity, its scalability and strong performance in optimizing feature extraction networks have led to its widespread adoption in many network extensions. An SE-block module is illustrated in Figure 3.6.

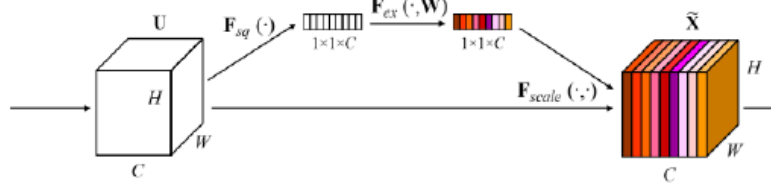


Fig. 3.6. SE block structure

An SE-block typically consists of two operations, Squeeze and Excitation. The Squeeze operation is used to obtain global feature information in the channel direction from the input feature U , which is achieved by introducing global average pooling (GAP) [30]. This transforms feature U from HWC to $1*1*C$, and the resulting feature vector after global average pooling has a global receptive field [31], which can be propagated to adjacent feature layers.

2) Using SENet to Improve Xception

After carefully analyzing SENet and Xception, it was decided to embed the SE-block into the residual structure of the Xception network, applying the SE-block to rescale features before they are added to the shortcut. This approach was chosen because when obtaining the channel weights in SENet through two fully connected layers, sigmoid is used to limit the weight range. If the SE-block were added after the addition operation, such weighting might result in the vanishing gradient problem in deep networks, causing the residual structure itself to lose its effect and leading to model degradation or convergence difficulties. The modified residual structure is shown in Figure 3.7.

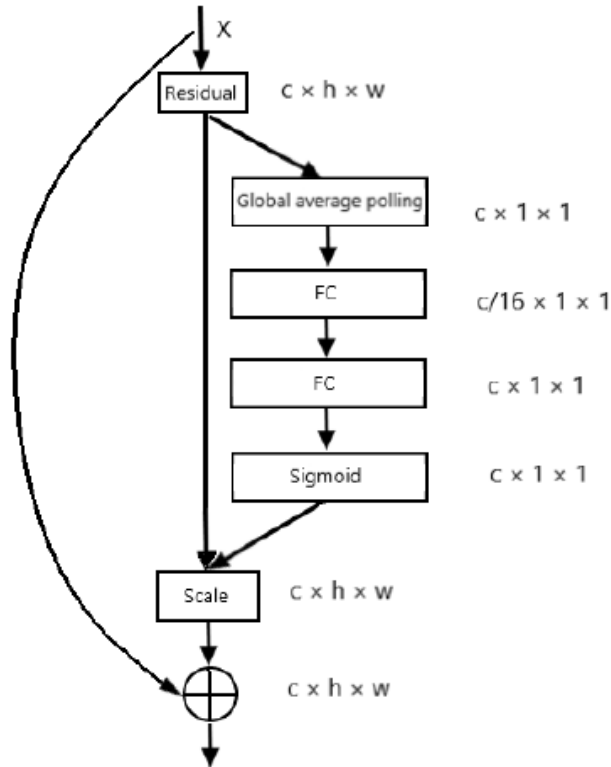


Fig. 3.7. Residual structure embedded with SE block e

IV. EVALUATION

A. Experimental environment and evaluation indicators

The machine configuration of this article is shown in Table 4.1.

Table. 4.1. The environment relied on in this article's experiment

CPU	Intel® Xeon(R) E5-2640 v4@2.4GH×20
Memory	32GB
Hard disk	1TB
GPU	NVIDIA GTX 1080ti × 2
operating system	Ubuntu 18.04.03 LTS
CUDA	10.2.89
cudnn	7.6.5
framework	tensorflow-gpu=1.14.0

The experiment was conducted using the Cityscapes dataset, which includes 2975 detailed annotated (gtFine) images of 18 cities in the training set and 500 in the validation set from three cities. The dataset contains 33 semantic classes for urban scenes, but considering practical factors, certain classes that are uncommon in ordinary cities, such as tunnels and trains, were excluded. Therefore, 12 specific classes of interest were selected as the experimental classification objects, and a label of 255 was used for classes that were not of interest. During the evaluation process, these pixels were not included in the loss function or miou calculation. While they were marked as white in the ground truth labels, it was hoped that the model could still independently learn the classification of these pixels.

The dataset was organized in tfrecord format and after labeling and tag conversion with scripts, the training and evaluation sets were converted to tfrecord format using TensorFlow. During training, data information was read through a TensorFlow queue with multiple threads, while a separate thread was responsible for reading the dataset, ensuring that data reading and GPU computation were performed simultaneously and reducing the idle waiting time of the GPU during training. For semantic segmentation evaluation, miou was used as the evaluation metric, representing the average intersection-over-union of predicted and ground truth values for each class. The miou value ranges from 0 to 1, with values closer to 1 indicating higher segmentation accuracy and better results.

$$IoU = \frac{TP}{(TP+FP+FN)} \quad (4-1)$$

$$MIoU = \frac{(IoU1+...+IoUn)}{n} \quad (4-2)$$

In this study, TP, TN, FP, and FN refer to true positive, true negative, false positive, and false negative, respectively. Specifically, TP represents instances where the model predicts a positive outcome and the actual outcome is also positive. TN represents instances where the model predicts a negative outcome and the actual outcome is also negative. FP represents instances where the model predicts a positive outcome but the actual outcome is negative, while FN represents instances where the model predicts a negative outcome but the actual outcome is positive.

B. Model training parameters

The hyperparameters used in the model training are shown in Table 4.2. For certain parameters in Table 4.2, an explanation is provided. The atrous_rate hyperparameter is a list-type variable that describes the dilation rate of the atrous convolution in the second to fourth layers of the ASPP structure. This parameter needs to correspond to output_stride. When atrous_rate is set to [12, 24, 36], output_stride should be set to 8, and when atrous_rate is set to [6, 12, 18], output_stride should be set to 16. This is because although using atrous convolution does not affect the final feature size, when the kernel size of the 3x3 atrous convolution is larger than the feature map size, it is effectively equivalent to a 1x1 convolution [32-33]. Therefore, in order to ensure that the atrous convolution works as expected, it is necessary to control the downsampling rate of the feature input to ASPP by controlling output_stride so that the size of the input feature equals the size of the dilated convolution kernel with the maximum dilation rate.

Table. 4.2. The hyperparameter used in this experiment training

num_clones	2
training_number_of_steps	30000
model_variant	xception_65
atrous_rate	6, 12, 18
output_stride	16
decoder_output_stride	4
train_crop_size	401,401
train_batch_size	8

dataset	cityscapes
init_learning_rate	1e-4
optimizer	momentum
momentum	0.9
learning_policy	poly
learning_power	0.9
initialize_last_layer	False
last_layers_contain_logits_only	True
fine_tune_batch_norm	False

The parameters `initialize_last_layer` and `last_layers_contain_logits_only` are a corresponding pair. The former parameter indicates whether to load all parameters when loading the pre-trained model. If the number of classes in the dataset remains unchanged, this parameter can be set to True to use the softmax parameters in the pre-trained model. However, because the number of classes in our experiments has changed, only the logits parameters of the model can be loaded during the loading process. Therefore, `initialize_last_layer` needs to be set as False, and `last_layers_contain_logits_only` needs to be set as True to ensure that the convolutional layers involved in classification are not included when loading the pre-trained model. In terms of learning rate decay, we adopt the poly strategy.

C. Timeline

Figure 4.1 displays a GANTT chart outlining the suggested project timeline.

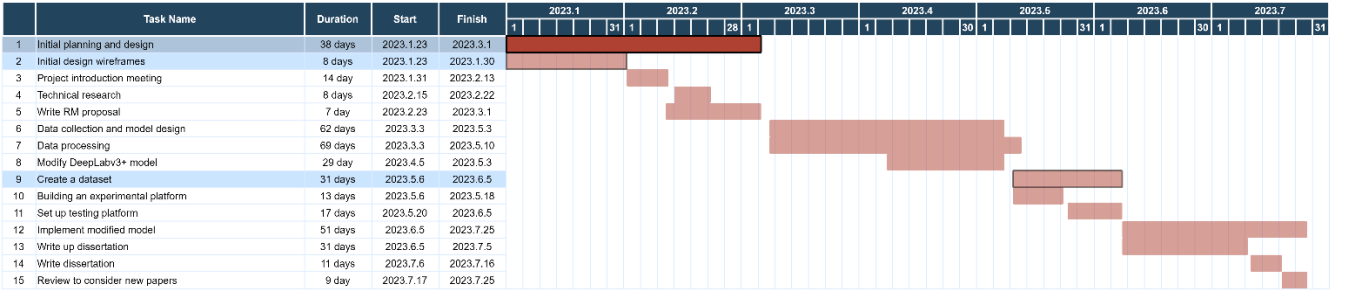


Fig. 4.1. project timeline

V. CONCLUSION

With the continuous development of surveillance devices, a large number of urban street view images have been generated. Currently, emerging technologies such as intelligent video quality analysis require precise semantic segmentation for accurate judgments by vehicle systems. Traditional semantic segmentation methods have a high reliance on prior knowledge, and they are subject to many constraints, which limits their robustness. These characteristics prevent their application in fields that rely on segmentation reliability, such as intelligent video quality analysis. In this paper, we propose an optimized segmentation network based on the DeepLabv3+ model [34-35], which performs better than the original DeepLabv3+ model on the Cityscapes dataset. The main contributions of this paper are as follows:

- (1) We introduced mainstream semantic segmentation networks based on deep learning, analyzed the innovations and limitations of these models, and trained and evaluated our model on the Cityscapes dataset by preprocessing it to set 12 interested categories and organizing the training data in tfrecord format.
- (2) We replaced the cross-entropy loss function in the original Deeplabv3+ model with a weighted focal loss function, addressing the problem of poor training results for small quantity classifications due to uneven classification numbers and training difficulties, thereby improving the model's miou on the evaluation set.
- (3) We modified the input of the Decoder by replacing the single shallow feature in the original model with three features from different layers of Xception_65 and using ASFF to adaptively fuse the three shallow features. To reuse parameters, the fused feature channels used the original model's 48 channels and concatenated the fused feature with the ASPP output in the channel direction, achieving improved segmentation accuracy.
- (4) We optimized the residual structure of Xception_65 using SE-block. Before the addition operation, we recalibrated the residual branch using SE-block and used the attention mechanism to strengthen important features and suppress secondary features, making the neural network more robust.

REFERENCES

- [1] Ullah W, Ullah A, Hussain T, et al. Artificial Intelligence of Things-assisted two-stream neural network for anomaly detection in surveillance Big Video Data[J]. *Future Generation Computer Systems*, 2022, 129: 286-297.
- [2] Vora S, Rangesh A, Trivedi M M. Driver gaze zone estimation using convolutional neural networks: A general framework and ablative analysis[J]. *IEEE Transactions on Intelligent Vehicles*, 2018, 3(3): 254-265.
- [3] Khosroshahi A, Ohn-Bar E, Trivedi M M. Surround vehicles trajectory analysis with recurrent neural networks[C]//2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2016: 2267-2272.
- [4] Ilyas N, Shahzad A, Kim K. Convolutional-neural network-based image crowd counting: Review, categorization, analysis, and performance evaluation[J]. *Sensors*, 2019, 20(1): 43.
- [5] Murugesan M, Thilagamani S. Efficient anomaly detection in surveillance videos based on multi layer perception recurrent neural network[J]. *Microprocessors and Microsystems*, 2020, 79: 103303.
- [6] Cordts M, Omran M, Ramos S, et al. The cityscapes dataset for semantic urban scene understanding[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 3213-3223.
- [7] Huang X, Cheng X, Geng Q, et al. The apolloscape dataset for autonomous driving[C]//Proceedings of the IEEE conference on computer vision and pattern recognition workshops. 2018: 954-960.
- [8] Ni Chenxu. A review of computer vision research. *Electronic World*, 2018(01):91-93.
- [9] Liu Shuo. A review of the development status of threshold segmentation technology. *Science and Technology Innovation and Application*, 2020(24):129-130.
- [10] Bu Wenbin, You Fucheng, Li Quan, et al. An image segmentation method based on the improvement of Otsu's method. *Journal of Beijing Institute of Graphic Communication*, 2015, 23(4):76-78.
- [11] Liu Jianzhuang, Li Wenqing. A 2D Otsu automatic threshold segmentation method for gray-scale images. *Acta Automatica Sinica*, 1993(01):101-105.
- [12] Jing Xiaojun, Li Jianfeng, Liu Yulin. An image segmentation algorithm based on three-dimensional maximum between-class variance. *Acta Electronica Sinica*, 2003(09):1281-1285.
- [13] Chen Fangxin. Image segmentation technology based on region growing method. *Science and Technology Information (Science Education Research)*, 2008(15):386-387.
- [14] Shen Xiaojiong, Wu Xiaoyang, Han Daojun. A review of watershed segmentation algorithm research. *Computer Engineering*, 2015, 41(10):26-30.
- [15] Guo Baolong, Hou Ye. An automatic image segmentation method based on graph cut. *Pattern Recognition and Artificial Intelligence*, 2011, 24(5). [16] Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation. arXiv:1411.4038v2
- [17] Simonyan K, Zissman A. Very deep convolution networks for large-scale image recognition.arXiv:1409.1556v6.
- [18] Badrinarayanan V, Kendall A, Cipolla R, et al. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp.2481-2495, 1 Dec. 2017.
- [19] Lin TY, Goyal P, Girshick R, et al. Focal loss for dense object detection. arXiv:1708.02002v2.
- [20] Hoh H, Hong S, Han B. Learning deconvolution network for semantic segmentation. arXiv:1505.04366v1.
- [21] Chen LC, Papandreou G, Kokkinos I, et al. Semantic image segmentation with deep convolutional nets and fully connected CRFs. arXiv:1412.7062v3.
- [22] Chen LC, Papandreou G, Kokkinos I, et al. DeepLab:semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. arXiv:1606.00915v2.
- [23] Chen LC, Papandreou G, Schroff F, et al. Rethinking atrous convolutional for semantic image segmentation. arXiv:1706.05587v3.
- [24] Chen LC, Zhu YK, Papandreou G, et al. Encoder-decoder with atrous separable convolution for semantic image segmentation. arXiv:1802.02611v3.
- [25] Cheng Lei, Wu Xiaofu, Zhang Suofei. Analysis of the impact of imbalanced datasets on transfer learning. *Signal Processing*, 2020, 36(1): 110-117.
- [26] Mikolov T, Kombrink S, Burget L, et al. Extensions of recurrent neural network language model. 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Prague, Czech Republic, 2011, pp. 5528-5531.
- [27] Lin TY, Goyal P, Girshick R, et al. Feature pyramid networks for object detection. arXiv:1612.03144v2.
- [28] Fu J, Liu J, Tian H J, et al. Dual attention network for scene segmentation. arXiv:1809.02983v4.
- [29] Hu J, Shen L, Sun G. Squeeze-and-excitation networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 7132-7141.
- [30] Lin M, Chen Q, Yan SC. Network in network. arXiv:1312.4400v3.
- [31] Luo WJ, Li YJ, Urtasun R, et al. Understanding the effective receptive field in deep convolutional neural networks. arXiv:1701.04128v2.
- [32] Yu F, Koltun V. Multi-scale context aggregation by dilated convolutions. arXiv:1511.07122v3
- [33] Wang Jingbin, Lai Xiaolian, Lei Jing, et al. Multi-scale atrous convolutional neural network model based on attention mechanism. *Pattern Recognition and Artificial Intelligence*, 2021, 34(6): 497-508.
- [34] Li Z, Wang R, Zhang W, et al. Multiscale features supported deeplabv3+ optimization scheme for accurate water semantic segmentation[J]. *IEEE Access*, 2019, 7: 155787-155804.
- [35] Wang Z, Wang J, Yang K, et al. Semantic segmentation of high-resolution remote sensing images based on a class feature attention mechanism fused with Deeplabv3+[J]. *Computers & Geosciences*, 2022, 158: 104969.