POLITECNICO DI TORINO

College of Electronic Engineering,
Telecommunications and Physics

Master's Degree Thesis

Waste Detection Based On Mask R-CNN

Supervisors
prof. Bartolomeo Montrucchio
dr. Antonio Costantino Marceddu

Candidate
Yushuo Chang

July 2022

Summary

The constant increase in the per capita production of household waste is accentuating the problem of their disposal and recycling. If disposed of improperly, they can cause serious damage to the ecological environment and pollute water, soil, and air. However, if waste is classified and treated correctly, the damage to the environment can be significantly reduced. The establishment of an automatic system for verifying the correct delivery of waste in the appropriate containers can improve the efficiency of recycling, promoting sustainable development.

In the past, researchers mostly used Support Vector Machines (SVMs) and others as the object detection algorithm. This required manual feature extraction and combination with the corresponding classifier. This method, for certain practical uses, can have low robustness and require a long training time. The advent of neural networks and in particular convolutional ones has made possible the advent of new object detection techniques based on deep learning. Under certain conditions, such algorithms may have greater operating accuracy and reliability than previously used techniques. Due to this reason, this thesis introduces and analyzes the characteristics of different R-CNN algorithms: the basic one, Fast R-CNN, Faster R-CNN, and Mask R-CNN. The latter model can classify and locate multiple objects with complex categories with great precision at the pixel level, making it suitable for the task wanted to perform.

The possibility of accurate analysis per pixel makes it possible to apply an instance segmentation algorithm. For its training, it was created through the use of the VGG Image Annotator (VIA) a dataset based on the COCO (Common Objects in Context) format containing 5175 different objects. It was obtained from 2451 different images of waste bins, provided by the company ReLearn. The final dataset was further split between training and validation datasets with a ratio of 8 to 2.

The Mask R-CNN model includes several parts: Feature Extraction Network (FEN), composed of the ResNet 101 neural network and Feature Pyramid Network, Region Proposal Network (RPN), used to extract the interesting areas contained in the image, ROI Align layer, which compared to the previous ROI Pooling improves the accuracy of the mask shape, and branch layer. In this last layer, three different activities are performed: classification, regression of the bounding box, and generation of the mask shape. With the same parameters, this thesis will analyze the variation in performance brought about by the variation in the size of the mask.

The following indicators were used to evaluate the model: Precision-Recall (PR) Average Precision (AP) of all categories, mean Average Precision (mAP), total loss,

and branch loss. By comparing these indices it can be concluded that the network having a mask size of 28 x 28 has the best performance. The respective mAP has reached a value equal to 82.85% and the Glass category has a maximum average accuracy of 86.71%. From the results of the prediction, it can be seen that this model is able to classify, in the same image, different categories of waste objects.

While the project has achieved impressive results, it could be further improved in several ways. One of them is to expand the size of the dataset. Through it, the neural network could learn to separate some characteristics of objects that can sometimes appear similar.

Acknowledgement

First of all, I want to give my sincere thanks to prof. Bartolomeo Montrucchio and dr. Antonio Costantino Marceddu who have helped me a lot in this thesis.

Especially, I want to thank Marceddu who have always given me useful suggestion and instructions throughout my work.

I am also grateful to all my friends and classmates who have kindly helped me when I have problems.

Finally, many thanks go to my family who gave me much encouragement and financial support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The first chapter introduces the background and significance of the thesis, briefly describes the current situation and development trend of automatic waste detection, and expounds on the innovation and main structure of this thesis.

1.1 Background

According to the world bank's research on waste management, there are nearly 4 billion tons of waste generated in the world every year, and urban waste alone accounts for a large proportion. It is expected that the amount of waste will increase by 70% in 2025. According to the data of [1], the waste accumulation in developing countries will increase significantly in the next 25 years. With the increase in the number of industries in urban areas, the treatment of waste has indeed become a concerning problem. Waste usually includes paper, plastic, metal, glass, and others. The main method of waste management is landfill, which has low efficiency, high cost, and pollutes the natural environment. It can also affect the health of the residents around it. Another common method of waste management is burning it in an incinerator. Like the previous one, it causes air pollution and can spread pollutants into the air, which increases the likelihood of contracting cancer [2]. In order to protect the environment, it is necessary to recycle and reuse the waste in different ways.

Knowing that a large part of the waste generated in large cities is recyclable, it is necessary to understand and apply reuse methods that can bring benefits or at least reduce environmental problems. The existence of technologies or models to help people sort waste is crucial to the proper disposal of this waste. Although there are different types of recycling categories, people are still confused or do not have a correct understanding of how to determine the correct garbage bin that can handle each type of waste.

Waste management and effective classification have been regarded as an important role in global ecological sustainable development. Society needs to reduce waste accumulation by recycling and reusing waste products. Effective sorting is usually used to improve recycling and reduce environmental impact. In developing

countries, waste management is a serious problem in their urbanization and economic development and should be dealt with in particular. In this sense, machine learning techniques can be a valuable aid. For this reason, this thesis will discuss a system based on Convolutional Neural Networks (CNN) and capable of detecting and classifying waste according to its type. Its training was made possible through a collaboration with the ReLearn company [28], which provided us with images of waste thrown into common bins. With its products, ReLearn aims to contribute to building a greener and more sustainable world. The images provided have been carefully labeled depending to their material: glass, metal, paper, plastic, trash, and compost. They have also been classified according to their type. These images were subsequently used in order to train a Mask-RCNN capable of verifying whether the waste is correctly classified and recycled. This system could be a valid aid to the improvement of recycling, increasing the value of knowledge and social stimulation in the classification and treatment of waste and improving community appeal to it. It can therefore have both a positive environmental and economic impact. For this purpose, this thesis will try to answer the following research questions:

- What type of deep learnin network is better to use in order to solve the problem of interest for the thesis itself?
- Can deep learning model effectively learn good feature representation from images to solve the waste sorting problem?

1.2 Related Models

Object detection (for more information see Section 2) is an important branch in computer vision. Many researchers are currently working on this problem to improve existing algorithms. Examples of these are like the works on target recognition [4] which can automatically recognize targets based on the data from sensors, image classification which can classify the object contained in the image [5]. Compared with simple target recognition or image classification, instance segmentation combines object detection and semantic segmentation, which is closer to the observation of objects with our human eyes and more suitable for trash detection. In 2012, AlexNet won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) with a Convolutional Neural Network (CNN) structure. Its introduction made traditional methods obsolete and led to a new era of deep neural network imaging [6].

In recent years, the rapid improvement of CNNs [7], has accelerated the development of object classification and detection methods. In 2014, Girshick and other authors proposed the Region-based Convolution Neural Network algorithm R-CNN (regions with CNN features) [8], which greatly improved the accuracy of object classification and detection. R-CNN first generates candidate windows, then carries out feature extraction, completes classification based on a Support Vector Machine (SVM), and finally runs a regression window. However, the detection efficiency of the R-CNN algorithm is low and takes up a lot of memory. Therefore, a short time later an improvement was proposed, called Fast R-CNN (fast regions

with CNN features) [9]. After inputting pictures into the network, it maps candidate windows to improve the speed of object detection. On the other hand, the network is optimized through adaptive pooling to improve the accuracy of detection. The problem with Fast R-CNN is that, since it adopts a selective search for the extraction of the characteristics of the candidate box, it has insufficient real-time performance. In 2016, again Ross B. Girshick and other authors proposed an improvement, called Faster R-CNN (faster regions with CNN features) [3]. In terms of structure, Faster R-CNN integrates feature extraction, bounding box regression, and mask generation into one network, which greatly improves the comprehensive performance, especially in terms of detection speed. Following the introduction of R-CNN, Fast R-CNN, and Faster R-CNN, a new algorithm was introduced, called Mask R-CNN. It can not only find the target object in the image, but it can also segment it accurately. Mask R-CNN has the following characteristics:

- Based on each candidate box in Faster R-CNN, an FCN (Fully Convolutional Network) [10] is used for semantic segmentation. Its powerful branch realizes the decoupling of the prediction relationship between mask and category. The mask branch only does semantic segmentation, and the task of category prediction is handed over to another branch.
- ROI Align is introduced to replace RoIPoooling in Faster R-CNN. Although it has little impact on the bounding box, it greatly improves the accuracy of the mask. After using ROI Align, the accuracy of the mask is significantly improved from 10% to 50% [3].

Based on many methods that can be applied to waste detection, Mask R-CNN exceeds all the end-to-end network models at that time, realizes pixel-level detection, and can accurately identify the contour of complex objects. In the practical application scenario of this thesis, objects with only local features and objects with different deformations will be included, and Mask R-CNN has a better detection effect than other algorithm models. Therefore, Mask R-CNN is selected as the algorithm model of this thesis.

1.3 Thesis structure

This thesis is divided in five chapters:

- Chapter 2 describes the basic idea and related works.
- Chapter 3 discuss about the database, models and methods used.
- Chapter 4 introduces the experimental results.
- Finally, Chapter 5 draws some conclusions.

Chapter 2

Related Works

The following chapter describes the related works, focusing on the current state of the art and the algorithms used in the models.

2.1 Object Detection

2.1.1 The History of Object Detection

The year 2012 marked a watershed for *object detection* techniques, which have largely shifted from traditional machine learning techniques to deep learning. After its successful introduction, the traditional object detection technologies were gradually replaced by the deep learning methods [11].

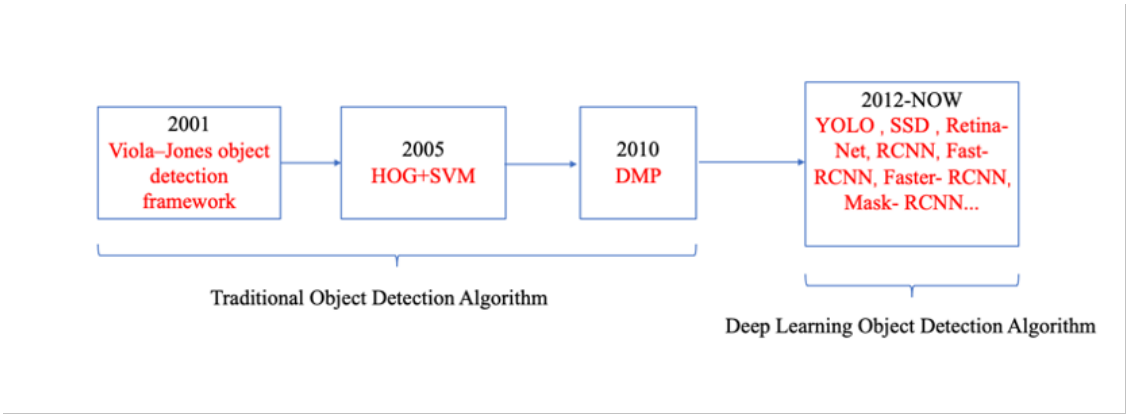

Figure 2.1. The development of object detection algorithm.

The development over the years of the object detection problem is shown in Figure 2.1. As shown in it, before the advent of Convolutional Neural Networks (CNNs) [12], object detection was based on traditional machine learning methods, with low detection accuracy and complex detection process. After the rise of CNNs, object detection based on deep learning has developed rapidly in recent years, gradually replacing the traditional object detection algorithms.

Traditional Object Detection	Deep Learning Object Detection
VJ detector, HOG+SVM, DPM ...	R-CNN, SPPNet Fast R-CNN, Faster R-CNN YOLO, SSD Mask R-CNN ...

Table 2.1. Traditional and deep learning methods.

Table 2.1 shows traditional object detection models versus deep learning models. At the moment, deep learning algorithms are emerging in endlessly, and the research on object detection in large Internet companies around the world is also progressing gradually.

2.1.2 Traditional Object Detection Algorithm

The traditional object detection algorithm is based on machine learning and its algorithm structure is simple. The overall flow chart is as follows:

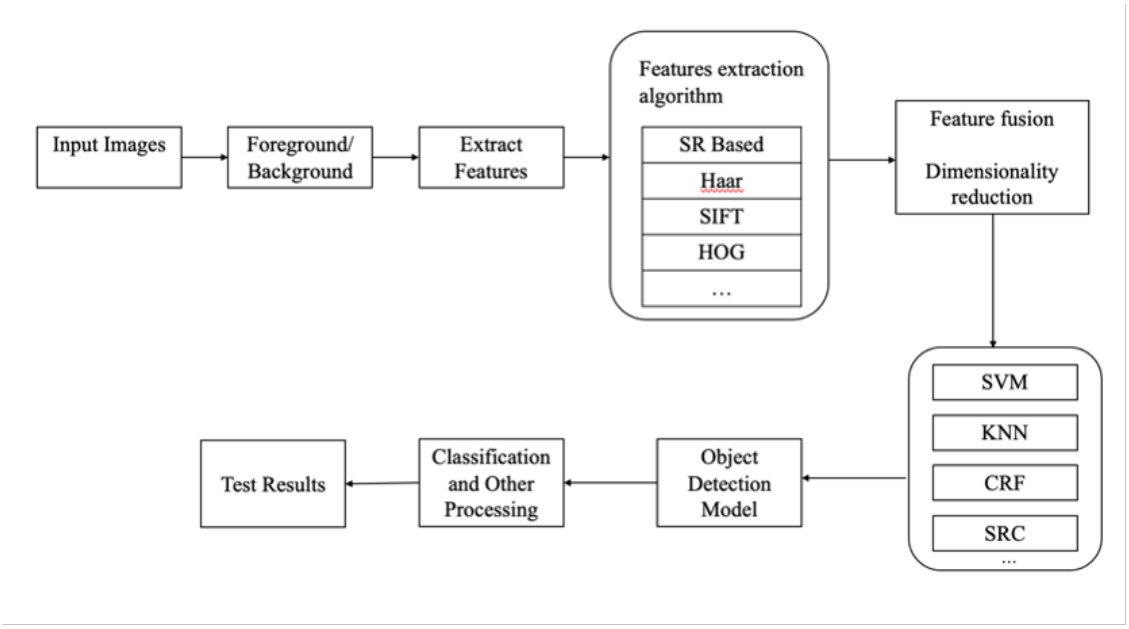

Figure 2.2. Flow chart of the traditional object detection model.

In 2001, P. Viola and M. Jones proposed a novel face detection algorithm [13]. Since computer hardware was not powerful enough at the time, it was developed to work even on devices with weak computing power. With the same detection accuracy, the detection speed was dozens or even hundreds of times that of other algorithms, laying a solid foundation for future face detectors. It was later called the "Viola Jones (VJ) detector".

In 2005, Dalia and Trigg proposed the Histogram of Oriented Gradient (HOG) [14], which is mainly used for vehicle and pedestrian detection. Usually, the HOG feature is combined with a Support Vector Machine (SVM) classifier [15]-[17]. Since HOG can capture local information, it has achieved great success in the field of

object detection. At present, most pedestrian detection algorithms use the idea of HOG + SVM for reference.

In 2010, Felenszwalb and others proposed a deformable part-based model (DPM [18]), which can be understood as splitting a detected object, detecting each of its parts, obtaining some local features, and then carrying out modular detection from part to whole. For example, to detect a bicycle, it is necessary to split each part of the bicycle and detect the handlebar, body, and wheel respectively. The features detected in these parts are fused to obtain the final detection results. The algorithm can also detect pedestrians. The detection process is whole-part-whole.

2.1.3 Object Detection Algorithm Based on Deep Learning

Before 2012, there was a bottleneck in the development of object detection, and the research on object detection was abandoned by researchers. After the great success that the CNNs had in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [19], object detection based on deep learning has become the focus of this kind of research.

In 2013, the introduction of the R-CNN detection algorithm broke the bottleneck of traditional object detection. This algorithm was proposed by Girshick and others [8] and marked the beginning of deep learning in object detection. It made CNN the core network of deep learning and increased their use in the field of artificial intelligence. The CNNs can be used for feature extraction and classification. The region proposal method has improved the accuracy of obtaining candidate frames, and many object detection frameworks have used this idea as a reference. The shortcomings of such a technique are low detection accuracy, low detection efficiency, and high resource consumption. In 2015, Fast R-CNN improved the detection accuracy, but detection speed was still slow, the detection efficiency was still low, and a lot of time redundancy was introduced in the detection process, making the algorithm unable to work in real-time.

In 2015, the new Faster R-CNN technique added the Region Proposal Network (RPN) [3] on the basis of the Fast R-CNN model, obtained the proposed regions through learning, improved the algorithm and so the accuracy and efficiency for obtaining the proposed regions. Compared with the speed of Fast R-CNN, the detection accuracy has also been greatly improved.

In 2016, the introduction of YOLO [20] and SSD [21] techniques realized end-to-end object detection [9]. With respect to Faster R-CNN, their models are quite different in structure. Compared with it, YOLO and SSD have greatly improved the detection efficiency and can perform real-time detection, but their accuracy is not as high as that of Faster R-CNN.

YOLO object detection model has been further improved with YOLO-V2 [22] and YOLO-V3 [23]. The improved model has achieved a better detection performance and better accuracy. To improve the object detection efficiency, the anchor points are dynamically obtained by the K-Means clustering algorithm.

In 2017, Kaiming He and his team, who at the time were part of the Facebook AI Lab, proposed the Mask R-CNN object detection model [24]. Aiming at the

problems of Faster R-CNN, such as low detection efficiency, the high missed detection rate for small targets, weak scalability for different data sets, and others, Mask R-CNN has brought an improvement in accuracy compared with Faster R-CNN. It can improve the shortcomings of Faster R-CNN in small object detection. it is also capable to implement instance segmentation.

In 2019, Kaiming He proposed PointRend: rendering idea for image segmentation [25]. Aiming at the problem that the edges of existing methods are not fine enough in the instance segmentation task, from the perspective of computer rendering, they proposed the PointRend method to better improve the smoothing and segmentation details in the image segmentation process.

2.2 Instance Segmentation

In past research, object detection and *instance segmentation* were seen as two different tasks. However, with the development of object detection technology, especially with the introduction of Faster R-CNN, the researchers found that both tasks can be performed simultaneously at the cost of a small increase in the amount of computation required.

The final scopes of object detection are as follows:

- Classification: solves the problem of understanding what the object is. This means when given a picture or a video, the model can analyze it in order to establish the class of the object it contains.
- Location: solves the problem related to determining the position of the object. The goal is to locate the actual location of the object.
- Detection: solves both the previous problems. So, it locates the position of the object and assigns a class to it.
- Segmentation: it is divided into instance level and scene level and tries to determine if each pixel of the image belongs to one or the other.

Compared with the simpler object detection, the result of instance segmentation brings more practical significance than the calculation cost: it can detect the objects and distinguish them in the overlapping state. The image instance segmentation serves to further refine the detection of objects, separating the foreground and background of the objects and realizing the classification of the objects at the pixel level. It can be applied in face detection, expression recognition, medical image processing, disease diagnosis, video surveillance, object tracking, shelf vacancy recognition in retail scenes, etc. As shown in figure 2.3, instance segmentation labels different objects of the same category, so it has high practical value.

Observing pictures c and d in the figure 2.3, it is possible to see that picture c is the result of semantic segmentation of picture a, while picture d is the result of instance segmentation of picture a. The biggest difference between the two is that the cubes in the figure are assigned the same color in semantic segmentation,

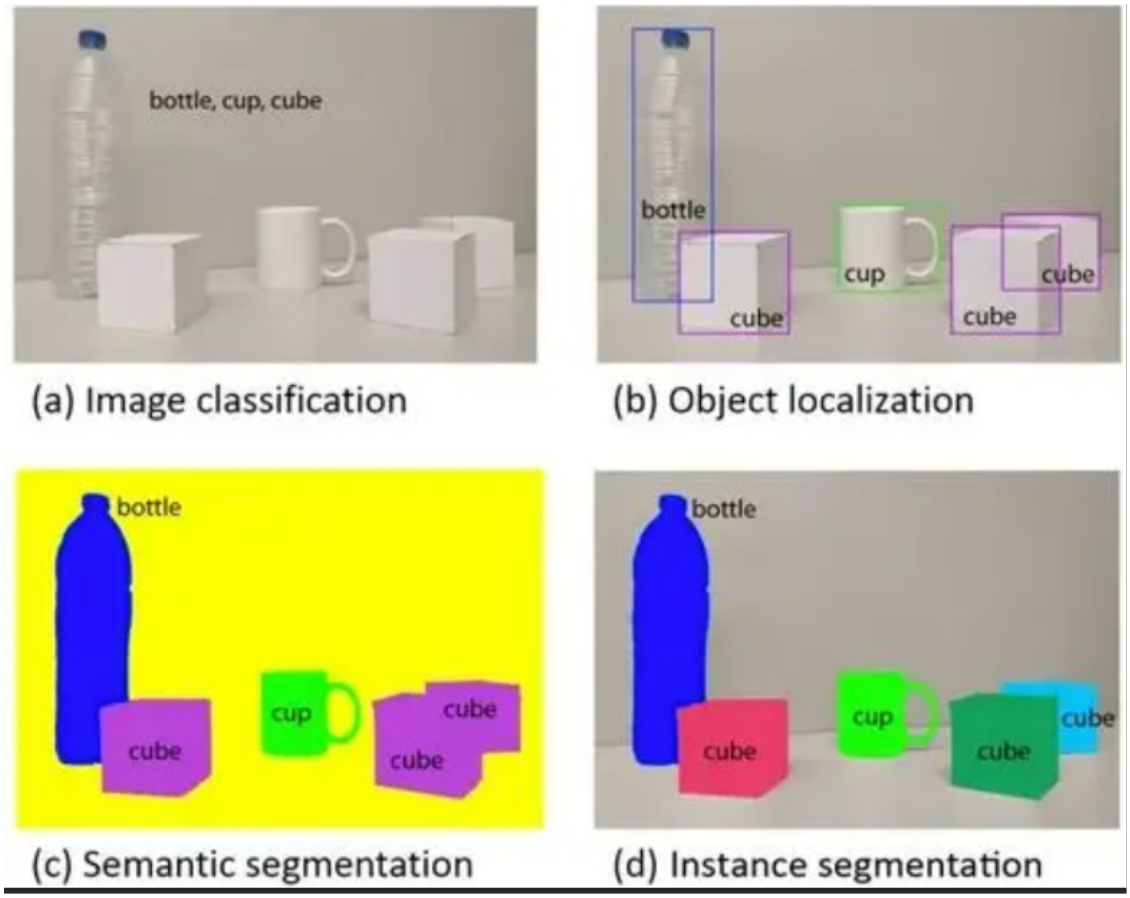

Figure 2.3. Example of instance segmentation [26].

but different colors in instance segmentation. That is, instance segmentation needs to perform a more accurate segmentation among similar objects than semantic segmentation.

As shown in Table 2.2, Mask R-CNN is one of the best instance partition networks at present.

Network Name	Feature Extraction Network	AP	AP50	AP75
MNC	ResNet-101-C4	24.6	44.3	24.8
FCIS	ResNet-101-C5	29.5	51.5	30.2
Mask R-CNN	ResNet-101-FPN	35.7	58.0	37.8
RetinaMask	ResNet-101-FPN	34.7	55.4	36.9
YOLACT-700	ResNet-101-FPN	31.2	50.6	32.8

Table 2.2. Mask R-CNN performance comparison [24].

It can be seen that the Average Precision (AP) of Mask R-CNN is higher than those of other networks. AP50 and AP75 are the AP values when the Intersection Over Union (IOU) of the prediction result and the object labeled box are 0.5 and 0.75, respectively.

2.3 Neural Networks

The *Convolutional Neural Network* (CNN) is a feed-forward neural network based on depth structure, which gets its name from the fact that it makes use of the convolution operation. The CNNs are mainly composed of convolution layers, pooling layers, activation layers, and fully connected layers. Compared with traditional neural networks, the CNNs have greatly improved generalization ability based on reducing model complexity. They can realize supervised and unsupervised learning at the same time, providing a new solution for tasks that were difficult to solve in the past.

2.3.1 Convolution Layer

The main function of the *convolution layer* is to extract the initial features of the image and generate its feature map. It is the key component of a CNN, which can contain one or more convolution layers. The convolution layer performs the convolution operation by using sliding windows on input images and feature maps. The main structural parameters include the following:

- The *stride* refers to the length of the convolution kernel shift through the input image. Its value is generally small so that fine information about characteristics can be stored in order to carry out, for example, detection and segmentation activities.
- The *convolution kernel size* determines the effective area of the convolution operation in the network. A large convolution kernel can be decomposed into several small convolution kernels. Sizes commonly used as a convolution kernel are usually 3x3 and 5x5. Furthermore, a 1×1 convolution kernel is widely used in lightweight networks because it can reduce the size of the feature map and reduce the computation of the model.
- The *padding* is designed to extract edge information around the image. It is designed to extract more information from the pixels present on the edges of the image. Convolution commonly tends to reduce the size of the output image. Another problem is that the values of the edges of the image are less important in the calculations. However, this problem can be partially avoided through the padding, which allows you to insert a specific pixel value beyond the boundaries of the image so that the size of the output feature map and input size can be the same. Padding values are usually 0 or equal to that of the closest pixel to the current one.
- The *input channel* defines the information enclosed by the input itself. For example, a grayscale image contains only the intensity channel, while an RGB image contains the red, green, and blue channels that combine to define every single color of every single pixel in the image.
- The *output channel* is obtained by applying the convolution operation to the entire input.

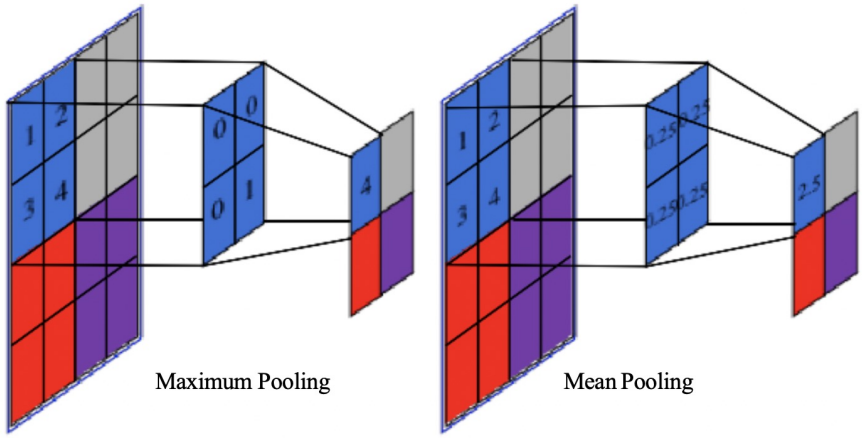

Figure 2.4. Pooling layer types.

During the process, the weight parameters on the convolution kernel and the pixels in the corresponding area of the window are multiplied and summed, and the offset coefficient is added to obtain the pixel value at the corresponding position of the output feature map. There are great differences in the features extracted from different convolution layers. Generally, the simple features can be obtained in the low-level network, while the rich complex features can be obtained in the high-level network. Assuming that the input image size is $W \times H \times D$, convolution kernel size is $F \times F$, the stride is S and the padding is P, the calculation formula of the output feature map size is the following:

$$\begin{cases} W^* = (W - F + 2P)/S + 1 \\ H^* = (H - F + 2P)/S + 1 \end{cases}$$

2.3.2 Pooling Layer

When the size of the output feature map in CNNs is too large, a huge number of high-dimensional features will be kept in the output results. If used directly in subsequent training, a large number of calculations will be produced, resulting in a slowing down of the learning process and an increased risk of overfitting. Therefore, to reduce the size of the feature map and improve the fitting ability, it is possible to introduce a pooling layer after multiple continuous convolution layers.

The main function of the *pooling layer* is to compress the pixel values in the sub-region of an image into a value, which can represent regional features and integrate neighborhood characteristics. Furthermore, the pooling operation has translation invariance and more. When the pixels of the input feature map undergo translation, rotation, and other transformations in the corresponding region of the pooled convolution kernel, the output value of the pooling layer will not be affected. As shown in Figure 2.4, pooling methods can be divided into two types based on the calculation method: average pooling and maximum pooling.

Mean pooling divides the image into smaller regions and takes the average value of all pixel values in each region as the value for the output position. In this process, the convolution kernel parameters are fixed and do not need to be updated iteratively through the back-propagation algorithm. After mean pooling, the feature map integrates all the features in the image and can play a certain role in the fusion of similar features.

Maximum pooling takes the maximum value of a pixel in a certain region in the feature map as the pixel value of the corresponding position after pooling. It reduces the estimation deviation caused by the complexity of convolution layer parameters, preserves, and extracts advanced features such as texture and background of the image, and highlights the differences between these different features.

The core function of the maximum pooling operation is the same as the mean pooling, which is to reduce the feature dimension of the feature map and reduce the resolution of the input image. The input feature map can also be filled with values equal to zero during the pooling operation to ensure that the image size remains unchanged before and after the pooling, and to be able to extract more feature information.

2.3.3 Activation Layer

CNN image processing through convolution or pooling layers is done through linear operations. However, in most of the images, data is generally nonlinear and inseparable. Therefore, it is necessary to resort to *activation functions* such that CNN obtains non-linear processing capacity and the modeling can adapt to more complex scenarios.

Assuming that a linear function is used as an activation function, no matter how high the number of layers of the network model is, the results obtained will still be equivalent to the linear combination of inputs. It cannot, therefore, approximate the complex function and the field of action is extremely limited and does not meet the requirements. The use of a non-linear activation function can instead help CNNs to learn more abstract characteristics and to improve the adaptability of non-linear factors. The most used activation functions are the Sigmoid, the TanH, and the ReLU, which will be introduced below.

2.3.3.1 Sigmoid function

The *Sigmoid* is an activation function commonly used in neural networks and its definition is given in the following formula:

$$\text{Sigmoid } (x) = \frac{1}{1 + e^{-x}}$$

As can be seen in Figure 2.5, the range of values of the Sigmoid function is limited between 0 and 1, which makes it suitable for estimating a probability. The Sigmoid is suitable for binary classification tasks as it has a smooth curve, a monotone continuity in scope, and an easy derivation. However, it also has the following disadvantages:

- For an input value away from the coordinate origin, the corresponding output value in the positive direction of the coordinate axis tends to be 1 and the output value in the negative direction tends to be 0. In this case, since the gradient is small, it is easy to make sure that the network parameters are not updated during the backpropagation training.
- The output value is always greater than 0, which is not symmetrical about the coordinate zero point. Also, the gradient always changes in one direction and the overall iteration rate of the network is slow.
- It presents an exponential operation, which has a high computational complexity.

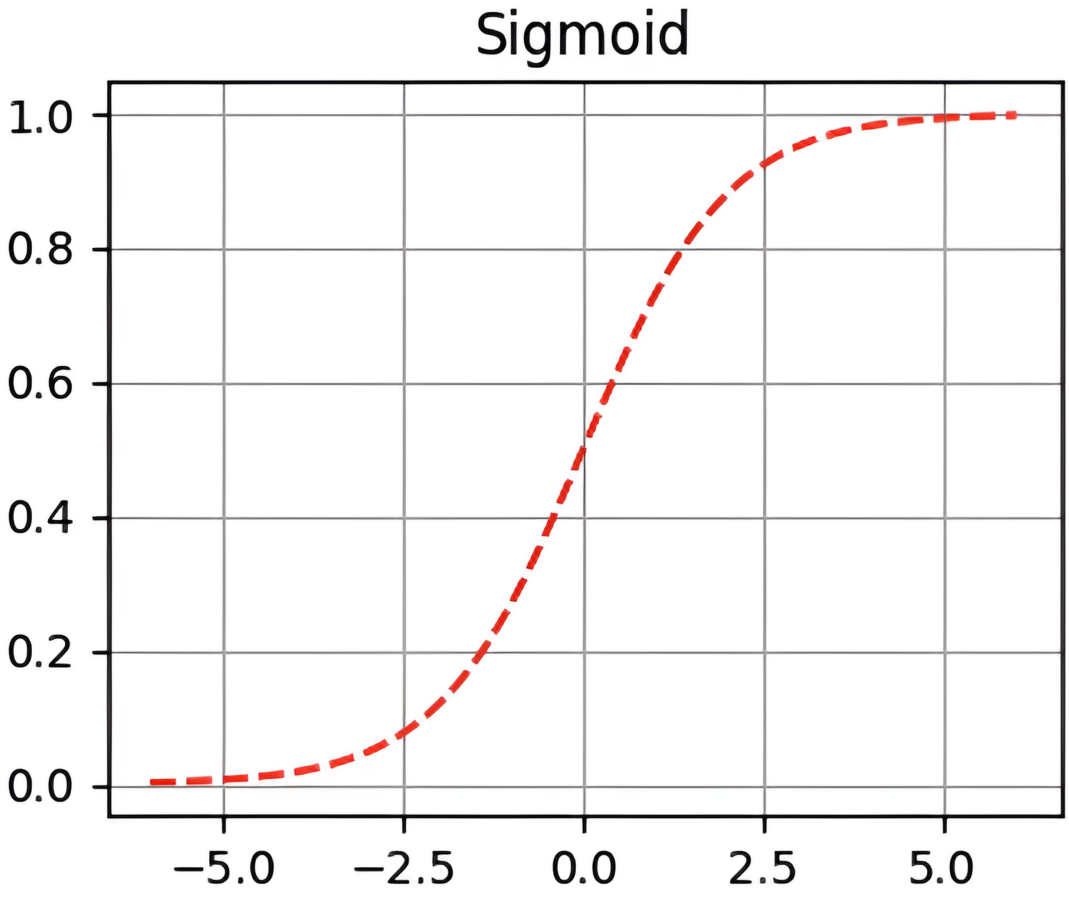

Figure 2.5. Activation function of the Sigmoid function.

2.3.3.2 TanH function

The *TanH* function is a variant of the Sigmoid function. Its definition is shown in the following formula:

$$\mathrm{Tanh}(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

As can be seen from Figure 2.6, the output value of the TanH function is between -1 and 1 and is symmetrical with respect to zero from the center. This solves the problem of the non-zero mean output of the Sigmoid function. Compared to it, the TanH function is very easy to train and has an improved adaptation speed. Its disadvantages are the following:

- The output at both ends of the function is relatively smooth and the gradient value is small. There remains the problem of the disappearance of the gradient due to saturation.
- The function uses exponential operation, and the derivative takes the form of a power exponent, which increases the amount of calculation and reduces the speed of network optimization.

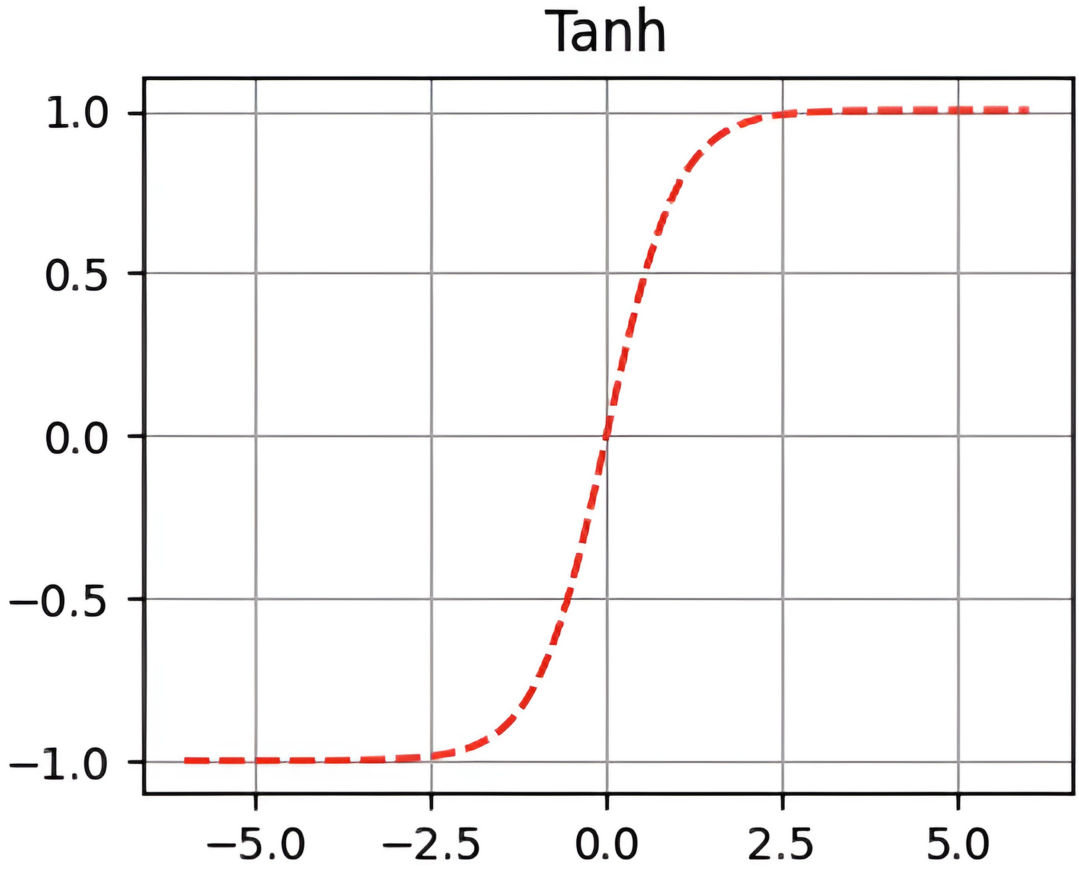

Figure 2.6. Activation function of the TanH function.

2.3.3.3 ReLU function

The *ReLU* function is a piecewise function and in some fields it is more used than Sigmoid and TanH. Its definition is given below:

$$\text{ReLU}(x) = max(0, x)$$

It can be seen from Figure 2.7 that the advantages of ReLU function are:

- When the input value is greater than 0, the derivative is equal to 1 and the gradient will not disappear as a result of updating the parameter parameters by backpropagation.
- No exponential operation is required and computational complexity is low.

However, when the input value is in the range $(-\infty, 0]$ the gradient of the function is always 0. This makes it impossible to activate the neurons afflicted by this problem and the weight value is not. Moreover, the output of the ReLU function, like the Sigmoid function, is not zero symmetric about coordinates.

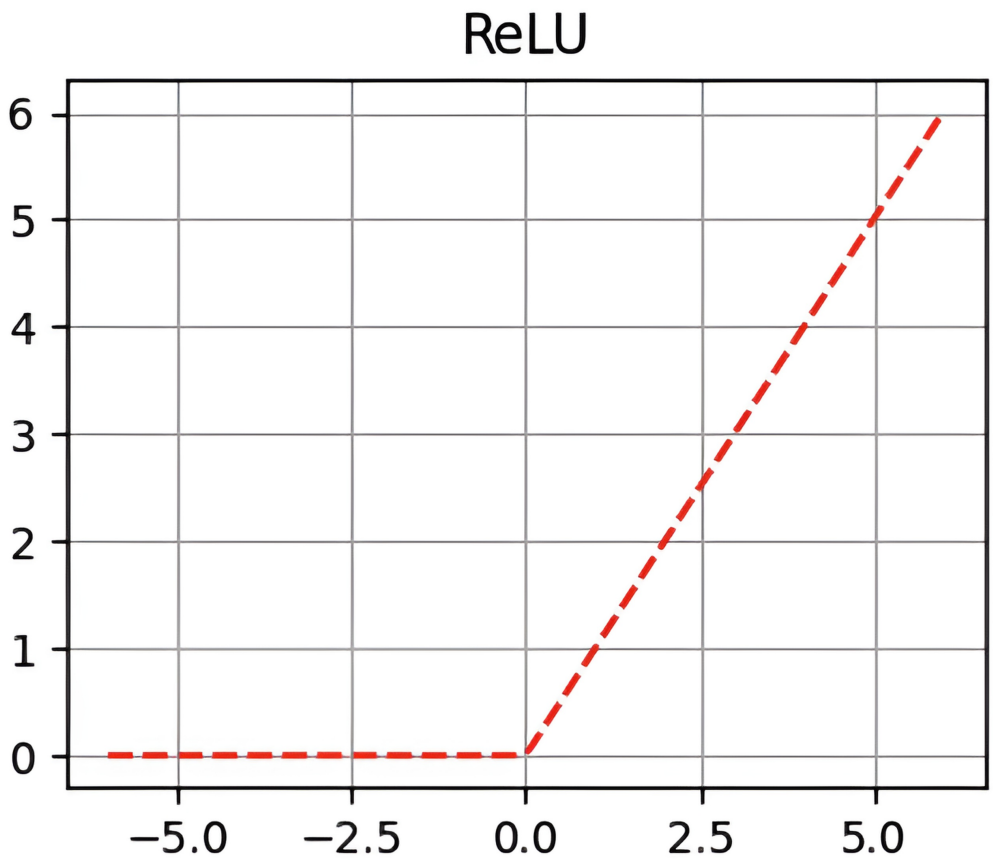

Figure 2.7. Activation function of the ReLU function.

2.3.4 Fully Connected Layer

The fully connected layer is generally used in the last layers of the CNN in order to prepare for the output results. An example can be seen in Figure 2.8. Its main function is to integrate the local features extracted through a series of convolution layers, pooling layers, and activation layers into the global features through the matrix of weight. The output of the fully connected layer can be inserted into the

objective function of the neural network, which can then perform the classification and regression activities to obtain the final output of the network model.

Unlike the convolution layer which has locally connected characteristics, the fully connected layer connects all neurons in the current layer with neurons in the upper layer. This increases the overall amount of computation required to train the network and ignores the spatial location information of the feature map. Therefore, in the specific design of a CNN, the size of the convolution kernel of the fully connected layer can also be 1×1. This structure can respond to different sizes of the input images without changing its spatial structure, which is more in favor of completing instance segmentation activities based on spatial information.

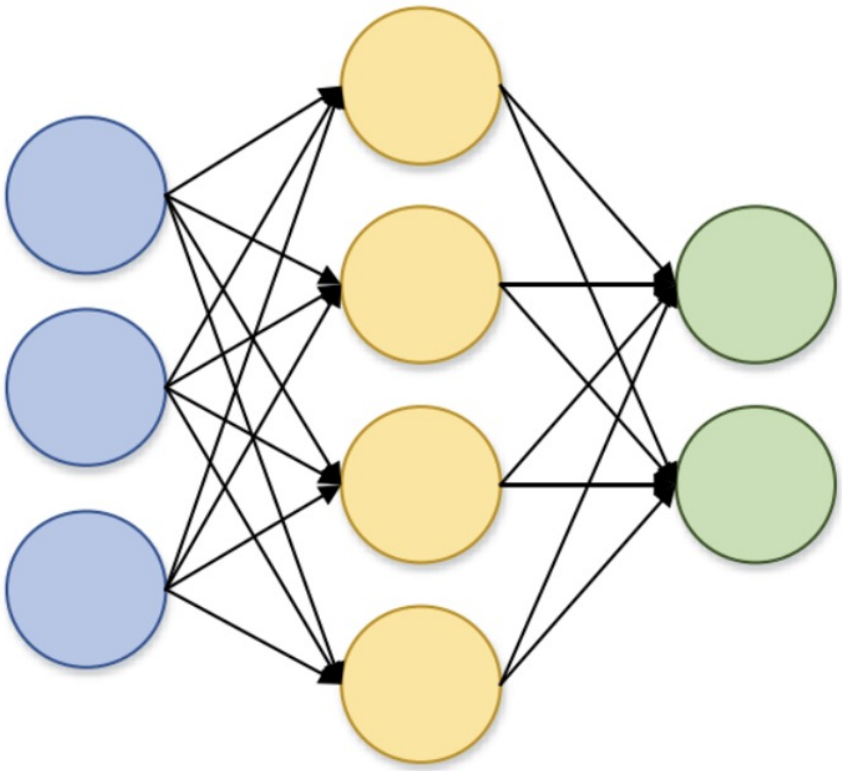

Figure 2.8. An example of fully connected layer.

2.4 R-CNN Algorithms

The *Region-based Convolutional Neural Network* (R-CNN) [8] was one of the first deep learning-based object detection models. The combination of a CNN with a selective search algorithm allowed it to perform well on the PASCAL VOC 2007 [32] dataset. Through simple bottom-up (for more information see Section 3.2.1) grouping, the selective search algorithm used by R-CNN can segment the input

image, calculate the similarity based on the features of color, size, and texture, merge regions of high similarity and generate the final regions proposed after a continuous iteration. The network structure of R-CNN is shown in Figure 2.9. First, a large number of independent proposed regions are generated in the input image using a selective search algorithm. These proposals, which may contain objects, are then fixed to a uniform size by clipping or deformation and are input to an AlexNet for the extraction of the functionalities. On this basis, multiple SVMs are used to complete the classification and the object prediction box is refined in combination with linear regression.

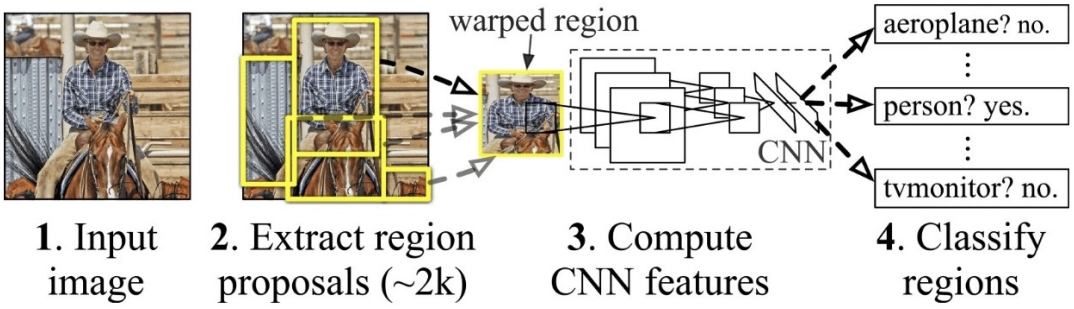

Figure 2.9. R-CNN structure [27].

Although the R-CNN algorithm shows a great improvement in accuracy compared with traditional object detection algorithms, it still has many defects. On the one hand, R-CNN transforms the scale of the proposed region to a fixed size, which will deform the proposed region and lose the original feature information. On the other hand, R-CNN needs to extract the features of all proposed regions one by one, which brings a huge computational cost, resulting in a very slow object detection speed. In addition, R-CNN is not an end-to-end network. It needs time-consuming training and wastes a lot of storage space, which makes it difficult to apply the algorithm to the industrial field.

2.4.1 Fast R-CNN Algorithm

Girshick, the original author of R-CNN, has continued to improve it with new additions. Referring to the network structure of *SPP-Net* (Spatial Pyramid Pooling Network) [9], he designed the *Fast R-CNN* algorithm [9]. The network structure of Fast R-CNN is shown in Figure 2.10. Its main innovation is that a *Region Of Interest pooling* (ROI pooling) is added to the fully connected layer and the previous convolution layer to extract the suggested features of each region. The feature maps of different sizes put into the layer are then merged to a fixed size and there is no need to crop the original image to meet the requirements of the fully connected layer. In this way, it is possible to preserve information on the spatial characteristics of candidate samples, reduce disk space occupation and improve training speed. In addition, Fast R-CNN introduces a multi-task loss function to train classification and regression tasks in parallel to accelerate model convergence.

The parameters calculated by the classifier do not need to be saved separately: which saves a lot of space and makes the implementation of the model easier.

The stream followed by the Fast R-CNN algorithm is as follows:

1. The proposed regions are generated using the selective search algorithm for the input image and CNN is used to extract image features.
2. Then, the ROI on the newly obtained feature map is inserted into the ROI pooling layer to form a unified feature vector.
3. Finally, a SoftMax classification is used as the output layer for multi-category classification. And bounding box regression are performed through the fully connected layer.

SoftMax function is often used in the last layer of the network as the output layer for multi-category classification.

Although the accuracy and speed of the Fast R-CNN algorithm have been greatly improved over R-CNN, there are still a number of limitations. One of these is that it is necessary to use a selective search method with a large amount of computation to obtain the proposed regions. Furthermore, the process of generating the proposed regions is still complex and inefficient. This method takes most of the training and prediction time and is the main limitation for further improving the speed of Fast R-CNN.

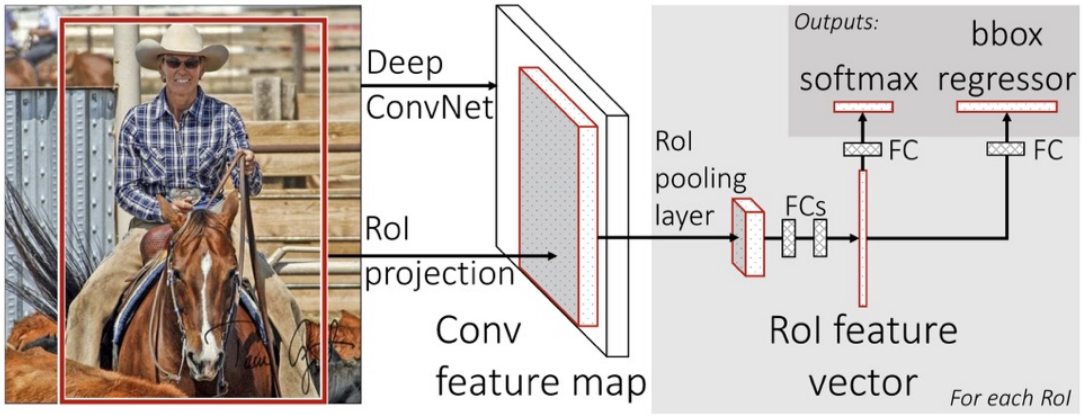

Figure 2.10. Fast R-CNN structure [9].

2.4.2 Faster R-CNN Algorithm

After extensive research, Ren proposed *Faster R-CNN* [3] in order to solve the problem related to the large computing resources required by Fast R-CNN. To improve model detection efficiency, it uses a *Region Proposal Network* (RPN) to replace the selective search method. The RPN is a full CNN structure, whose main

function is to generate high-quality proposed regions through end-to-end training. In Faster R-CNN, the convolution feature map of the input image is inserted in the RPN and in the detection branch at the same time and this saves a lot of computing resources. As shown in Figure 2.11, the network structure of Faster R-CNN is mainly composed of three modules: feature extraction, RPN, and classification regression.

The flow of the Faster R-CNN algorithm is as follows:

1. The CNN is used to get the feature map corresponding to the input image.
2. Then, the RPN is used to tentatively classify the foreground and background and generate proposed regions.
3. Like Fast R-CNN, the ROI pooling layer is used to produce a fixed-size feature map.
4. Finally, the classification confidence score of the object is obtained in the classification branch, and the positioning of the coordinate of the object is carried out in the regression branch.

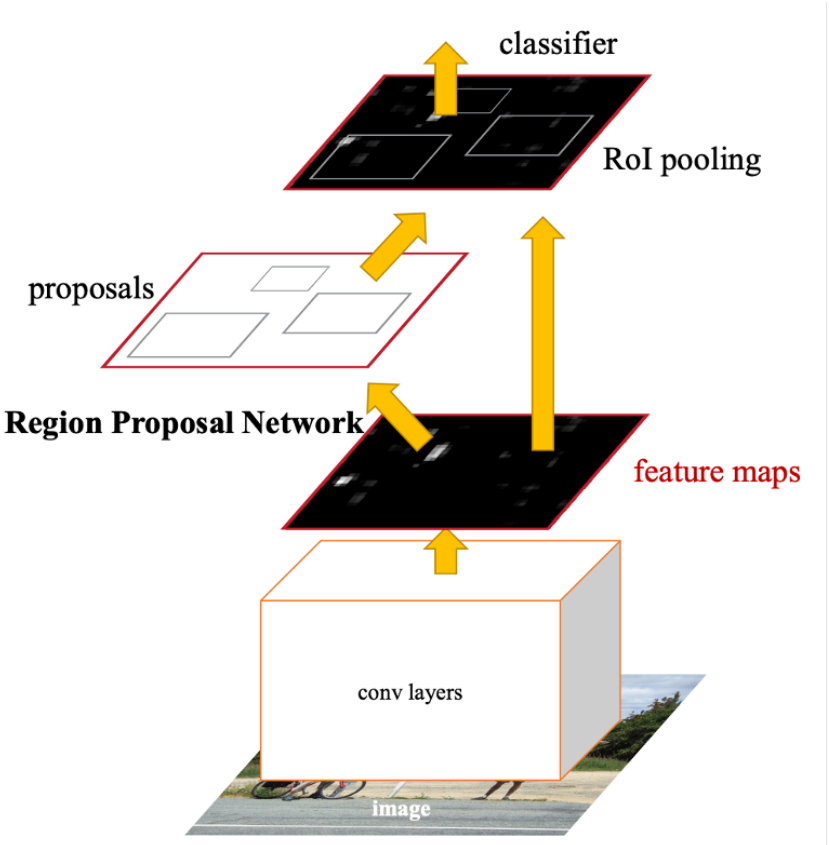

Figure 2.11. Faster R-CNN structure [3].

Chapter 3

Proposed Solution

In this chapter the implementation of the proposed system will be presented in detail. It includes data processing, training model creation and prediction result.

3.1 Dataset Preparation

The object detection model usually contains a large number of parameters which, during the training process, are calibrated in order to capture the features of the objects involved. This process is not possible if there is not a finely classified dataset. In this regard, this section will discuss in detail how the trash image dataset was made.

3.1.1 Data Collection

The images used to create the dataset were kindly provided by *ReLearn* [28], a company which aims to contribute to building a greener and more sustainable world. Through a device under development, called *Nando*, it was possible to acquire images of waste present in real bins. They were the basis for the creation of the dataset necessary for training the instance segmentation system. The dataset includes six types of materials for waste:

- Metal.
- Plastic.
- Paper.
- Glass.
- Metal.
- Trash.

Figure 3.1. How Nando works [30].

It also includes a classification for the object type, such as plastic bottle, paper bag, tin cans and others. It has not been used in this thesis as the number of images available does not make it possible to use it. Due to the confidentiality agreement which was stipulated with the company it is not possible to publish any photos of the dataset. However, alternative images from a public dataset on the Internet will be used to illustrate the dataset creation process and testing [29]. It is possible to see some of them in Figure 3.2.

Figure 3.2. Example of images containing waste used in the thesis itself [29].

3.1.2 Data Annotation

The dataset provided by ReLearn is made up of 2451 images in .jpg format having a size of 1024 × 768 pixels. The dataset images have been split into training subsets and validation subsets with a ratio of 8 to 2. The exact number is shown in Table 3.1. When training a neural network that can distinguish between the above classes, it is necessary to ensure that the images in the dataset are representative and evenly distributed. Furthermore, the image data in the training dataset and validation dataset should not be repeated, so as not to interfere with the experimental results.

Dataset	Number of images
Train data	1968
Validation data	492

Table 3.1. Subdivision of the images used between training and validation datasets.

VGG Image Annotator (VIA) [31] was used to label the dataset. It is an annotation tool written in the Python language which does not require any installation or configuration and which is capable of running in a web browser. It can be used to annotate various objects and generate mask images. The VIA software interface is shown in Figure 3.3.

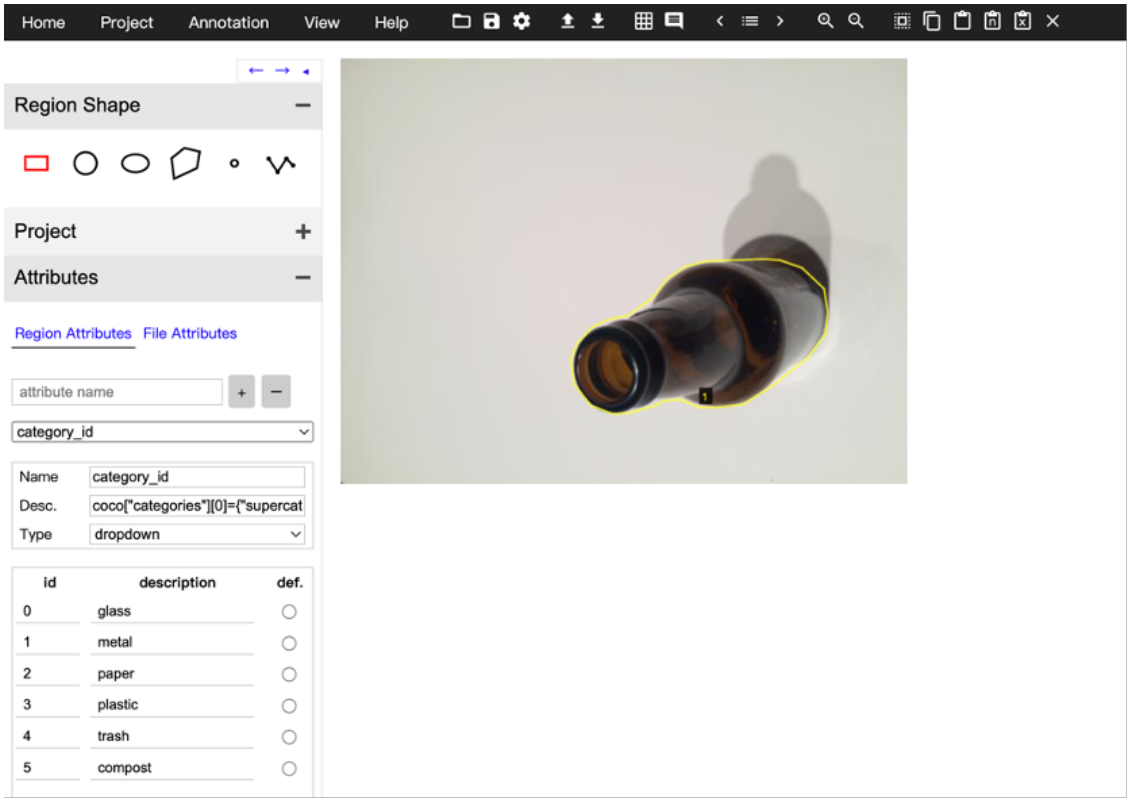

Figure 3.3. VGG Image Annotator interface [29].

Its use is quite simple and is based on the following steps:

- Import the region attributes file.
- Open the image file.
- Select the appropriate shape of the toolbar on the left side of the screen.
- Divide each region of the object in the image by connecting it into lines.
- Assign preset labels or define a new one to label the image.

As shown in Figure 3.4, the preset labels include the category ID and the object ID.

```
1 {"category_id":
2     {"0":"glass","1":"metal","2":"paper","3":"plastic","4":"trash",
3     "5":"compost";}
4 "object_id":
5     {"0":"bottle","1":"bowl","2":"chopstick",..."28":"paper cpu";
          }
6 }
```

Figure 3.4. Preset labels.

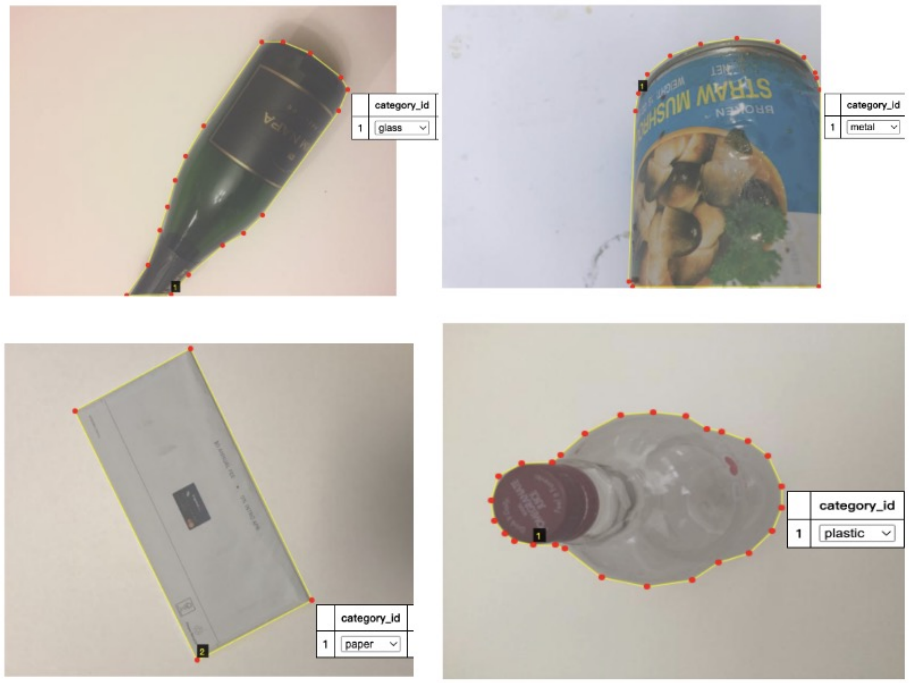

Figure 3.5. Example of labeled images [29].

Examples of the labeled dataset can be seen in Figure 3.5. By saving the annotations it is possible to obtain a single .json file in which all the necessary information is present. An example of this file is shown in Figure 3.6. It is possible to obtain the region of a single object from the x and y coordinates and from the object categories, which are very important in the subsequent training process.

```
{"filename":"2022-02-22___16-04-33.jpg",
    "size":, //the size of image
    "regions"
    {"shape_attributes":
        {"name":"polygon",
        "all_points_x":[...], //labeled region
        "all_points_y":[...]}},
        "region_attributes":
    {"category_id":"2", //the object category
    "object_id":"25"}
},
"file_attributes":},
```

Figure 3.6. An example of the contents of the .json file.

3.1.3 Data Format Convert

Before training the Mask R-CNN, it is necessary to convert the format of the labels. Various formats are commonly used for them, such as Pascal Visual Object Classes (VOC) [32] and Common Objects in Context (COCO) [33]. In this thesis, the latter was chosen as the preferred format. Specifically, COCO is a large-scale object detection, segmentation, key point detection, and subtitle dataset. Its format is shown in 3.7.

```
COCO_ROOT    #root directory
    ├── annotations    #store annotations in json format
    │     ├── instances_train2017.json
    │     └── instances_val2017.json
    └── train2017        #store image files
    │     ├── 000000000001.jpg
    │     ├── 000000000002.jpg
    │     └── 000000000003.jpg
    └── val2017
          ├── 000000000004.jpg
          └── 000000000005.jpg
```

Figure 3.7. COCO dataset format.

All the annotations are stored in one .json file. It has a dictionary data structure, including the following five key-value pairs: info, images, licenses, annotations, and categories. Info and licenses are the default value when making our own dataset. Only three fields are needed:

- *images*, which are a list of dictionaries that store the file name, height, width, and ID of the image. The ID is the unique number of the single image and is also used in annotations. There are as many dictionaries in the list as the number of the image.

```
# json['images'][0]
 {
  'file_name': '2022-02-27___09-26-33.jpg',
  'height': 768,
  'width': 1024,
  'id':0
 }
```

- *categories*, which refers to all previously defined categories. The category ID starts from 1 while 0 stands for background. The format is as follows:

```
0:
    id:1
    name:"glass"
    supercategory:"none"
1:
    id:2
    name:"metal"
    supercategory:"none"
...
5:
    id:6
    name:"compost"
    supercategory:"none"
```

- *annotations*, which refers to the annotations of the labeled region. The format of such a region is as follows:

```
0:
    id:0
    image_id:0
    category_id:4
    object_id:13
    iscrowd:0
    area:917.999999999999
    bbox:[...]
    segmentation:[...]
...
5175:
    id:5175
```

```
image_id:239
category_id:3
object_id:20
iscrowd:0
area:2483.9999999999995
bbox:[...]
segmentation:[...]
```

In it, id indicates the id of the single bbox, image _id reports the id of the image corresponding to the unique serial, category _id and object _id report the ids of the category and the corresponding object, area indicates the area segmented, bbox indicates the coordinates [x, y, w, h] of the detection box and segmentation indicates the segmented polygon. The more bboxes there are, the more dictionaries will be present in the annotations.

Several functions have been created in the via2coco.py file in order to convert the data format:

- create_image_info creates image dictionaries.
- create_annotation_info creates annotation files, including segmentation information.
- convert loads the original information and links the image id with segmentation info and categories, then generates the training and validation dataset in COCO format.

The final datasets used for the traning are depicted in Figure 3.8.

Figure 3.8. The dataset obtained, further divided between training and validation datasets.

3.2 Mask R-CNN Model

The network structure of *Mask R-CNN* is shown in Figure 3.9. It is mainly composed of the following modules:

- Feature Extraction Network (FEN), responsible for calculating the multi-scale convolution features on the full image.
- Region Proposal Network (RPN), which generates the rectangular box of proposed regions.
- Region Of Interest (ROI) Align layer, which pools the generated proposed regions.
- Branch layer, which gives out the prediction category, bounding box, and object mask.

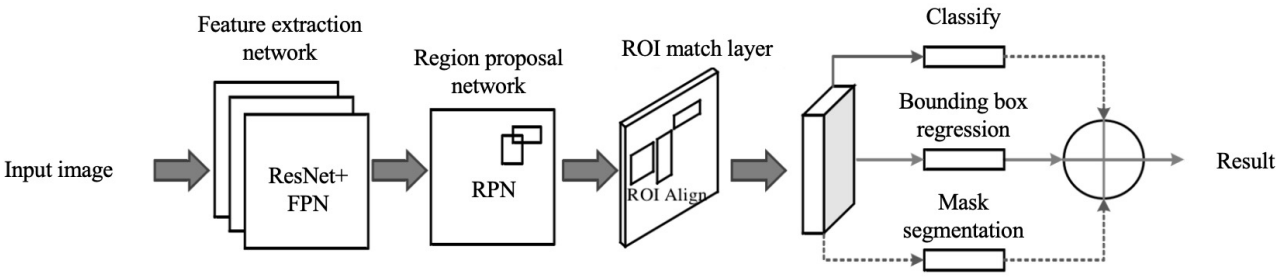

Figure 3.9. Structure of Mask R-CNN instance segmentation network.

The overall flow of the algorithm is as follows: first, the image containing waste is inserted into the instance segmentation network of the Mask R-CNN, the FEN extracts the characteristics of the image containing waste, and the generated feature map is inserted in the RPN to obtain the proposed regions. After non-maximum suppression, the size of the feature map is unified through the ROI match layer. In the output layer, the position coordinates and corresponding categories of the candidate bounding boxes are determined using the classification branch and the regression branch of the bounding box. Finally, in the segmentation branch, the binary mask of all the waste objects present in the image is predicted using the FCN, thus obtaining the segmented image.

3.2.1 Feature Extraction Network

The *Feature Extraction Network* (FEN), used in this thesis is composed of ResNet101 [34] as the backbone network and Feature Pyramid Network (FPN), which will be introduced below.

3.2.1.1 ResNet101

Compared to the initial phase, the development of CNN has led to an increase in the depth of the network model. They can increase the number of features

extracted and potentially increase model performance. However, in the actual training process, the researchers found that even if they use a deeper model, the improvement in the performance of the model is less noticeable, or even worse. This is because the network must be calibrated according to the complexity of the problem. Furthermore, the training of a deep network frequently involves problems such as overfitting, disappearance of the gradient, and degradation of nonlinear expression ability. The traditional solution is to increase the number of training samples, adjust the model parameters, and change the number of iterations, which is both time-consuming and resource-consuming as well as inefficient. The ResNet proposal [34] provides a good idea to solve this problem. Its main design idea is based on the fact that CNNs have the ability of identity mapping. The functional expression of identity mapping is shown in the following formula:

$$\mathrm{H}(x) = x$$

This means that the result of the output is the same as the original input. When CNN has the ability of identity mapping, it can ensure that the results upstream and downstream of the convolution are consistent in the process of stacking network layers. The main feature of the residual network is that the original input is transmitted to the output after the multilayer convolution by means of a quick connection. This implies a breakdown of the limitation concerning the fact that the output of the previous level can only be used as the input of the next level in the previous CNNs like AlexNet [6], VGG [35] and others, with no additional parameters required. The overall functional expression of each residual module in the residual network is shown in the formula below, of which identity mapping is an important part:

$$\mathrm{H}(x) = F(x) + x$$

As shown in the formula below, the important goal of the network layer in the training process is to fit the residual function F(x):

$$\mathrm{F}(x) = H(x) + x$$

The advantage of this design is that even if the number of layers of the residual network is very high, it is difficult to cause it to degrade as long as it is not limited by software and hardware conditions. Also, the difficulty of residual learning is less than learning the original output, which can accelerate model convergence.

Taking the two-layer structure as an example, the residual module design in the ResNet network is shown in Figure 3.10 below, while the expression of the corresponding function is shown in the following formula:

$$\mathrm{y} = F(x, \{Wi\}) + x$$

where x is the input of the residual block, y is the output of the residual block, and $\mathrm{F}(x, \{Wi\})$ is the final residual to learn. The definition of the objective function F is shown in the following formula:

$$\mathrm{F} = W_2\sigma\left(W_1 x\right)$$

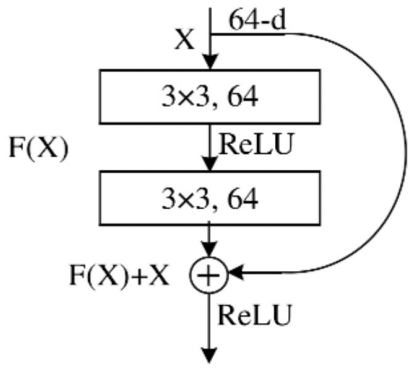

Figure 3.10. Residual block.

In it, σ represents the ReLU activation function, while W_1 and W_2 represent the learning parameters corresponding to the network layer. In this structure, the size of the convolution layer is generally 3×3.

In the design of the network structure, in order to reduce the difficulty of optimizing the model, ResNet adopts a residual block structure with three layers that are well suited to deep networks. It is based on the two-layer structure, also known as the bottleneck structure, and is shown in Figure 3.11. In this structure, the convolution part in the residual block is mainly composed of layers with dimensions of 1×1 pixel and 3×3 pixels. The 1×1 convolution can be used to reduce the size of the input feature map and the number of channels of the feature map to avoid unnecessary computational consumption. On the other hand, it can expand the number of channels of the output feature map to the size of the input feature map. Compared with the two-layer residual structure, the three-layer bottleneck structure can greatly reduce the number of model parameters and improve the network training speed.

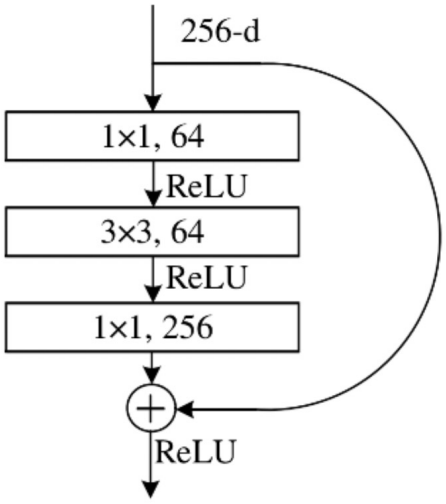

Figure 3.11. The bottleneck structure.

3.2.1.2 Feature Pyramid Network

As shown in Figure 3.12, in this thesis the feature extraction branch adopts the *Feature Pyramid Network* (FPN) network architecture. The FPN structure mainly includes three parts: bottom-up, top-down and lateral connection.

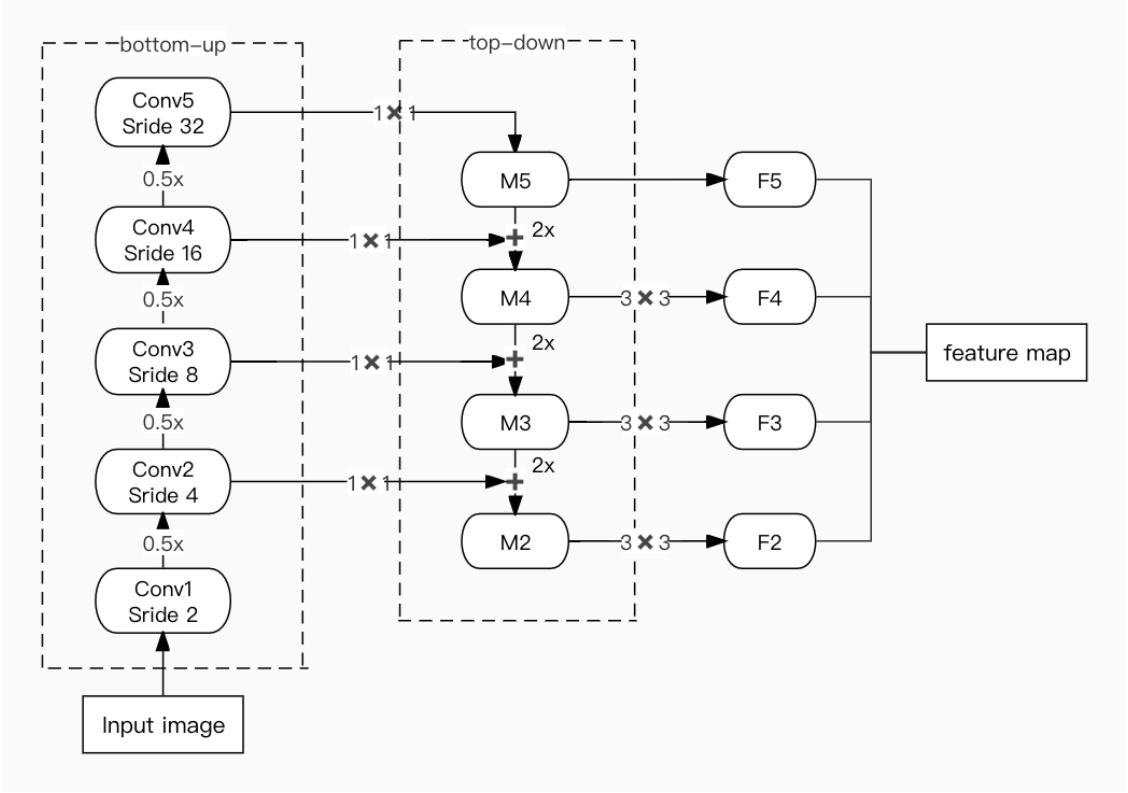

Figure 3.12. A FEN based on FPN.

Bottom-Up *Bottom-up* is the process of inputting images into backbone ConvNet to extract features. The size of the feature map output from the backbone is either unchanged or reduced by 2 times. Taking ResNet as an example, the outputs of convolution blocks Conv2, Conv3, Conv4, and Conv5 are defined as $\{C_2, C_3, C_4, C_5\}$, which are the outputs of the last residual block in each stage. These outputs are $\{\frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}\}$ times of the original graph respectively, so the relationship between the sizes of these feature maps is of two.

Top-Down The process of *top-down* process consists of upsampling the feature map obtained from the high level and then transferring it downwards. This is because high-level features contain advanced semantic information, which can be spread to low-level features via top-down propagation so that even low-level features contain advanced semantic information. In this thesis, the sampling method used is *nearest neighbor upsampling* (interpolation), which causes the feature map to expand twice.

The purpose of the nearest neighbor upsampling is to enlarge the image. Based on the pixels of the original image, a suitable interpolation algorithm is used to

insert new pixels between the pixels. This is the simplest interpolation method, which does not require calculation. Among the four adjacent pixels of the pixel to be calculated, it is assigned the value of the nearest adjacent pixel to be calculated. Its coordinates are set as $(i+u, j+v)$, i, u are positive integers and j, v are decimal greater than zero and less than 1, then the value of gray of the pixel is calculated as $f = (i+u, j+v)$. The top-down approach is shown in Figure 3.13.

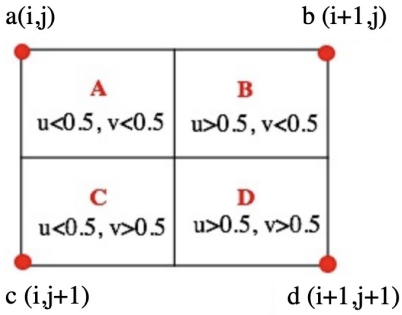

Figure 3.13. Nearest neighbor upsampling.

If $(i+u, j+v)$ falls in area A, i.e. $(u < 0.5, v < 0.5)$, the pixel value of point A is assigned to the pixel to be calculated. Similarly, if it falls in area B, the pixel value of point B is assigned to the pixel to be calculated. The amount of computation required by the nearest neighbor method is small, but it may cause discontinuities in the gray level of the image generated by interpolation. Moreover, jagged shapes may appear where the gray level changes.

Lateral Connection As shown in Figure 3.14, the *lateral connection* mainly includes three steps:

1. For the output of the C_n feature map coming from each stage, a 1 x 1 convolution is first performed in order to reduce its size.
2. Then, the obtained features are merged with the F_{n+1} feature map obtained by sampling the upper layer, i.e. by direct addition. Since the relationship between the feature maps emitted by each stage is twice, the size of the feature map sampled from the previous layer is the same as this layer, so corresponding elements can be added directly.
3. After the addition, a 3 x 3 convolution is required to get the feature output F_n of this layer. Convolution eliminates the aliasing effect caused by upsampling. This effect coincides with the problem of the jagged shapes mentioned above.

3.2.1.3 Network Parameters

ResNet101 model is used here to build a deep FEN and the extracted features are output through four convolution layers. The FPNs can extract multi-scale features,

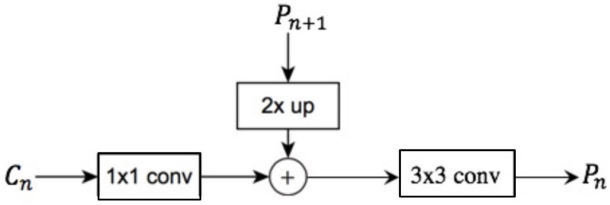

Figure 3.14. Lateral connection.

and the tracking performance of multi-object can achieve very good stability, especially for targets with large changes. It extracts the low-level functionality through Conv1 and gets the deeper functionality of different sizes through the convolution levels ranging from Conv2 to Conv5. The higher the number of layers, the richer the semantic information contained. The four-size feature maps obtained are fused through the upsampling operation to obtain the feature map of the object.

Network Layer	Parameters	Stride	Output
Conv1	$7 \times 7, 64$	1×1	128×64
MaxPool1	3×3	2×2	64×32
Conv2	$\begin{pmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{pmatrix} \times 3$	1×1	64×32
Conv3	$\begin{pmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{pmatrix} \times 4$	1×1	32×16
Conv4	$\begin{pmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{pmatrix} \times 6$	1×1	16×8
Conv5	$\begin{pmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{pmatrix} \times 3$	1×1	8×8

Table 3.2. Parameters of the network model.

Table 3.2 shows the parameters of each network layer of the ResNet model in the FEN, including the size of the convolution layer, the number of channels, and the output size.

3.2.2 Region Proposal Network

The *Region Proposal Network* (RPN) is a FCN. It performs a sliding convolution on a series of feature maps obtained from the feature extraction layer, then general proposed regions through the category less convolution network. Since the network adopts a two-class classifier, it will distinguish if there are proposed regions containing waste objects, calculate the coordinates of the center point, the length, and width of the input image which corresponds to the ROI of each of them and determine the coordinate position of the bounding box. The RPN structure is shown in Figure 3.15.

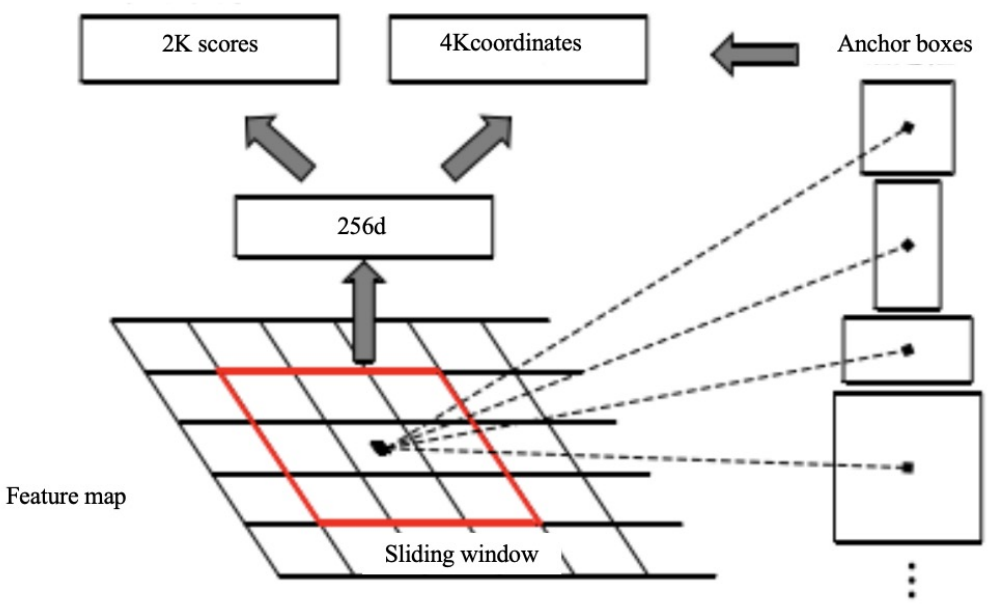

Figure 3.15. FEN based on FPN.

The specific operational steps are as follows. First, the input convolution feature map is passed to an n × n convolution kernel to perform the sliding convolution, generating the feature vector of the full connection layer corresponding to the sliding window. At the same time, considering the center of the sliding window, anchoring boxes are generated with each movement. The length-to-width ratio and size of the anchor box are preset and are related to the scale and aspect ratio. Each of them can get the corresponding region proposals. For a convolution feature map of size W × H there will be W × H × k anchor boxes. Figure 3.16 shows an example of the anchor boxes generated on an image containing waste.

After the feature vector is generated by the sliding window, two branches are connected. One is the classification branch, which is used to determine whether the proposed area is the foreground or the background. The other is the regression branch. The output value is the offset from the original coordinates, which is used to predict the coordinates x, y, and w, h (weight, height) corresponding to the central anchor of the proposed region. During training, only when the proposed region is classified as foreground and IoU is greater than the threshold, it will be

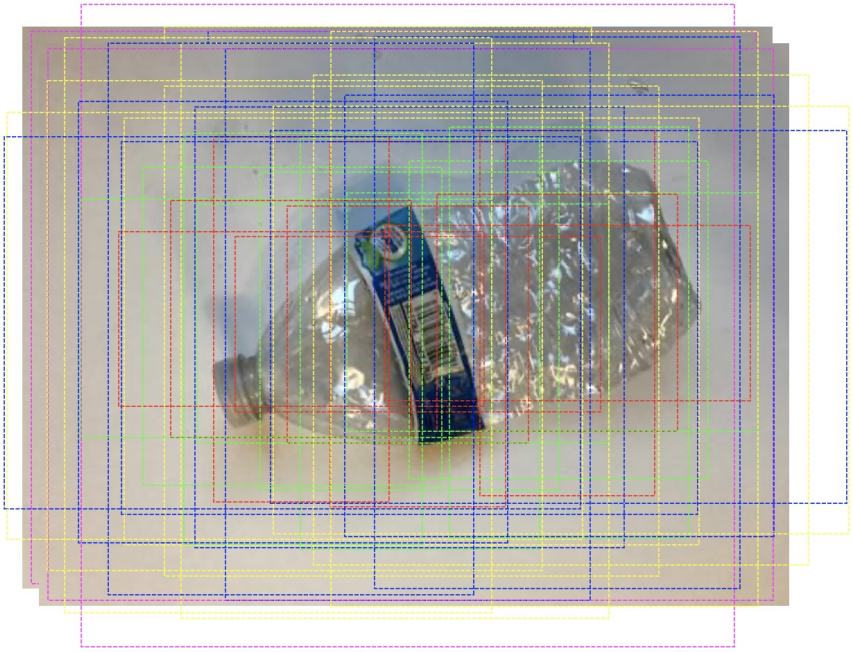

Figure 3.16. Example of generated anchors [29].

divided into positive samples, and then it will be subject to bounding box regression to determine the position of the candidate boxes and output foreground scores. In Figure 3.17 the IoU representation is depicted, while its calculation method is shown in the following formula:

$$IoU = \frac{Area_A \cap Area_B}{Area_A \cup Area_B}$$

A is the candidate box output by the region proposal network, B represents the object box correctly labeled in the trash dataset, A∩B is the intersection area of the candidate box, and A∪B is the union area.

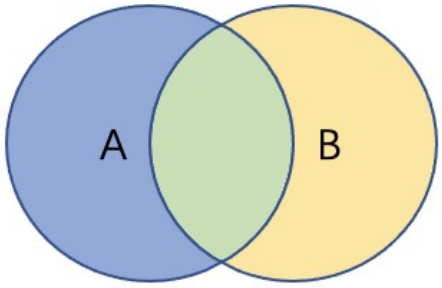

Figure 3.17. IoU representation.

The RPN will divide all generated anchors into positive, neutral, and negative

anchors by the early introduced branches. When there are too many anchors overlapped, the RPN only chose the one with the highest foreground score. By doing so, ROIs are obtained. In Figure 3.18 shows the anchor selection process.

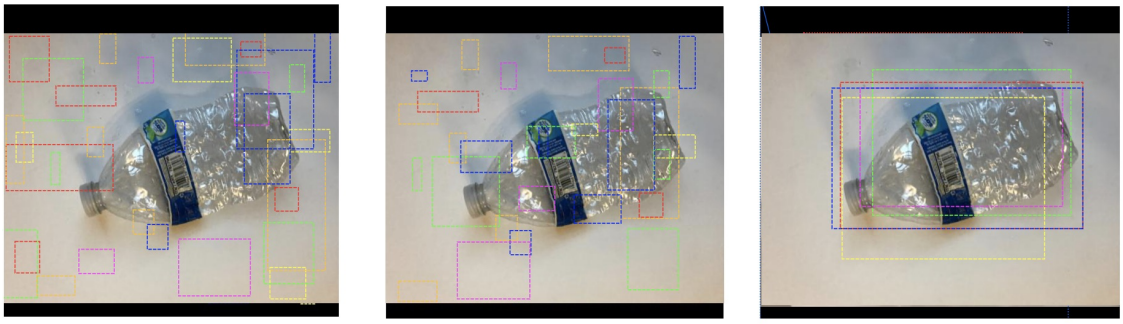

Figure 3.18. Example of the anchors selection process [29].

3.2.3 ROI Align

The RPN layer can produce a large number of overlapping candidate boxes. It is necessary to use the *Non-Maximum Suppression* (NMS) algorithm it is possible to exclude more accurate candidate boxes with high foreground confidence, and then put them together with the output feature map in the *ROI Align* layer, which improves the output of the ROI pooling layer.

Although the main purpose of ROI Align and ROI pooling is to get the candidate box with a fixed size, their implementations are very different:

- In ROI pooling, the position of the candidate box is obtained by regression and, for the most part, floating-point numbers are obtained. It is then necessary to correct the size of the feature map, so the ROI pooling operation has two quantization processes, resulting in misalignment between the original image pixels and the feature map. This leads to a large deviation in the ROI mapping of the feature space from the original map, resulting in an impact on the instance segmentation task at the pixel level.
- The ROI Align layer pools the ROI of the proposed regions generated in the regional information aggregation mode in Mask R-CNN in order to fix the feature maps of different scales in a unified scale. The Mask R-CNN model selects the bilinear interpolation method in the ROI Align layer to calculate the coordinates so that the originally discrete pooling process is continuous. In this way, pixel values that use floating-point numbers as coordinates can be mapped without any quantization. This method largely solves the region calibration problem caused by two quantifications in the ROI pooling process, better meets the requirements of the instance segmentation task, and achieves a more accurate position of feature points, improving the precision of the instance segmentation model.

Figure 3.19 shows the ROIs after the ROI Align layer.

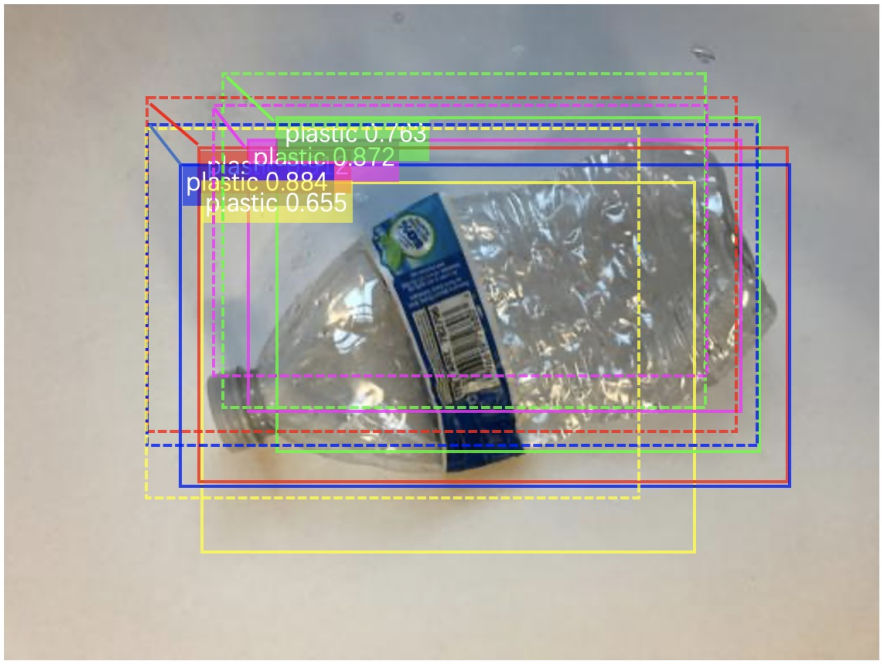

Figure 3.19. ROIs after the ROI Align layer [29].

3.2.4 Loss

The *loss* function is an estimate of the gap between the prediction result and the real label. The smaller the value of the loss function, the greater the robustness of the model. Learning iterations are essentially the process of minimizing the loss value. In Mask R-CNN, the total loss function is mainly composed of three parts: the classification loss of candidate boxes, the bounding box regression loss, and the mask loss. The specific definition is shown in the following formula:

$$Loss = L_{cls} + L_{bbox} + L_{mask}$$

3.2.4.1 Classification Loss

L_{cls} represents the loss due to the categories. This term refers to the foreground (containing the waste) and background (not containing the waste) and not to the output classes. This loss can exclude those boxes whose prediction category is background in the process of generating candidate boxes, and improve the precision of the Mask R-CNN model in predicting candidate boxes. The calculation method is shown in the formula below:

$$L_{cls} = \frac{1}{N_{cls}} \sum_{i} -\log\left[p_i^* p_i + (1 - p_i^*)\,(1 - p_i)\right]$$

p_i represents the probability of candidate boxes to be predicted as a positive sample with serial number i, N_{cls} represents the normalization parameter, $p_i^* = 0$ represents

the fact that the proposed region is a negative sample, and $p_i^* = 1$ represents the fact that the proposed region is a positive sample.

3.2.4.2 Bounding Box Loss

L_{bbox} represents the regression loss of the bounding box, which aims to make the bounding box coordinates predicted by Mask R-CNN close to the real values labeled in the input image. For the background, the expected coordinate correction has no value, so the bounding box regression is not performed. The calculation method of L_{bbox} is shown in the formula below:

$$L_{box} = \frac{1}{N_{bbox}} \sum_i p_i^* R\left(t_i, t_i^v\right)$$

$$\text{Smooth}_{L1} = \begin{cases} 0.5X^2 & \text{if}|X| < 1 \\ |X| - 0.5 & \text{otherwise} \end{cases}$$

N_{bbox} is the normalized parameter, t_i is the predicted offset parameter, t_i^v is the actual offset parameter, $p_i^* = 1$ and $p_i^* = 0$ represents the fact that the proposed region is positive and negative samples respectively, and finally R is the $Smooth_{L1}$ loss, as shown in the formula above.

3.2.4.3 Mask Loss

L_{mask} is the mask loss, which is added in the segmentation task, and the average binary cross-entropy loss is selected. This loss helps Mask R-CNN classify each pixel and promotes the instance segmentation. The calculation method is shown in the formula below:

$$L_{mask} = -\frac{1}{m^2}\left[\sum y_v \log y_v^k + (1 - y_v) \log\left(1 - y_v^k\right)\right]$$

y_v is the real tag value of the object and y_v^k is the predicted value in Mask R-CNN. Given that K is the number of object categories of the instance segmentation task, the mask output dimension corresponding to each region of interest is $K \times m \times m$. The network will predict the binary value of each pixel in the $m \times m$ size feature map, then judge whether the mask belongs to a certain category. When measuring the error, only the loss of its corresponding category is considered and the loss of other categories is not calculated. The specific category of the mask is determined by the classification branch, which leads to the decoupling of mask segmentation and classification. This is different from the semantic segmentation method in FCN. FCN classifies each pixel at the same time, which will lead to competition between categories.

3.3 Training Process

3.3.1 Experimental Environment

In the experimental process of this thesis, a unified platform is used. Python was chosen as the programming language, while TensorFlow and Keras were chosen as the main development framework for the deep learning part. The characteristics of the hardware used to perform the training are shown in Table 3.3, while the software used and their respective versions are shown in Table 3.4.

Hardware name	Configuration description
CPU	Intel Xeon E5-2683 v3
Memory	32G
Graphic memory	11G
Hard disk	2TB

Table 3.3. Characteristics of the hardware used for the training.

Software name	Version
Python	3.6
TensorFlow	1.14.0
Keras	2.2.5
Cuda	10.0

Table 3.4. Information on the software used in conjunction with their respective versions.

3.3.2 Hyperparameters Setting

An important step in training the instance segmentation model on trash images is configuring the *hyperparameters*. Unlike convolution kernel parameters which are automatically updated by the back-propagation algorithm, hyperparameters are model parameters that are manually preset before training the neural network. This can have a big impact on the training progress of the model.

In the experimental process, except for the estimation of training parameters based on experience, it is important to adjust the hyperparameters over time based on the training performance, in order to speed up the process and increase the robustness of the model obtained. Deep learning network hyperparameters include, for example, *learning rate* and *number of epochs*.

Among them, the setting of the learning rate is very important. Not only does it determine the model training rate, but a correct choice determines the success or failure of good model convergence. It is possible to choose to leave it fixed or to

```
Configurations:
BACKBONE                       resnet101
BACKBONE_STRIDES               [4, 8, 16, 32, 64]
BATCH_SIZE                     2
BBOX_STD_DEV                   [0.1 0.1 0.2 0.2]
COMPUTE_BACKBONE_SHAPE         None
DETECTION_MAX_INSTANCES        100
DETECTION_MIN_CONFIDENCE       0.7
DETECTION_NMS_THRESHOLD        0.3
FPN_CLASSIF_FC_LAYERS_SIZE     1024
GPU_COUNT                      1
GRADIENT_CLIP_NORM             5.0
IMAGES_PER_GPU                 2
IMAGE_CHANNEL_COUNT            3
IMAGE_MAX_DIM                  1024
IMAGE_META_SIZE                19
IMAGE_MIN_DIM                  800
IMAGE_MIN_SCALE                0
IMAGE_RESIZE_MODE              square
IMAGE_SHAPE                    [1024 1024    3]
LEARNING_MOMENTUM              0.9
LEARNING_RATE                  0.001
LOSS_WEIGHTS                   {'rpn_class_loss': 1.0, 'rpn_bbox_loss': 1.0, 'mrcnn_class_loss': 1.0, 'mrcnn_bbox_loss': 1.0, 'mrcnn_mask_loss': 1.0}
MASK_POOL_SIZE                 14
MASK_SHAPE                     [28, 28]
MAX_GT_INSTANCES               100
MEAN_PIXEL                     [123.7 116.8 103.9]
MINI_MASK_SHAPE                (56, 56)
NAME                           coco
NUM_CLASSES                    7
POOL_SIZE                      7
POST_NMS_ROIS_INFERENCE        1000
POST_NMS_ROIS_TRAINING         2000
PRE_NMS_LIMIT                  6000
ROI_POSITIVE_RATIO             0.33
RPN_ANCHOR_RATIOS              [0.5, 1, 2]
RPN_ANCHOR_SCALES              (32, 64, 128, 256, 512)
RPN_ANCHOR_STRIDE              1
RPN_BBOX_STD_DEV               [0.1 0.1 0.2 0.2]
RPN_NMS_THRESHOLD              0.7
RPN_TRAIN_ANCHORS_PER_IMAGE    256
STEPS_PER_EPOCH                1000
TOP_DOWN_PYRAMID_SIZE          256
TRAIN_BN                       False
TRAIN_ROIS_PER_IMAGE           200
USE_MINI_MASK                  True
USE_RPN_ROIS                   True
VALIDATION_STEPS               50
WEIGHT_DECAY                   0.0001
```

Figure 3.20. The chosen hyperparameters.

gradually decrease it during the training course. The latter often helps smooth out the training curve and speed up convergence.

The number of epochs instead refers to the number of training iterations for all data samples. The number of epochs to achieve model convergence is often positively correlated with the amount of data and the complexity of the model. The more complex the model and the more data samples there are, the more epochs are needed. In this thesis, the instance segmentation network model is trained on the self-created waste dataset. The hyperparameters settings used during training are shown in Figure 3.20. It is possible to see that:

- The initial learning rate was set to 0.001.
- The batch size was set to 2.
- The number of steps per epoch was set to 1000.
- The number of epochs was set to 100.

3.3.3 Mask Branch

During the training process, the mask branch is used to extract the local features corresponding to different categories. The workflow is shown in Figure 3.21. In the mask branch, the input information is ROI selected by the upper layer, then the features extracted from the FEN are stored as a mask Matrix with a fixed size,

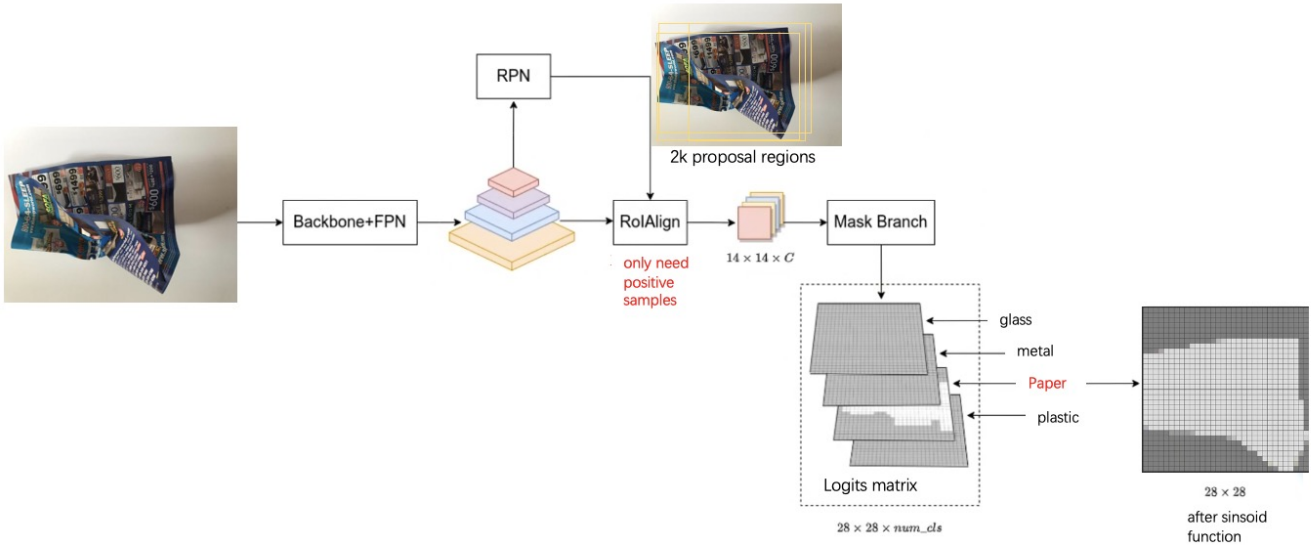

Figure 3.21. Mask branch during the training process [29].

for example 28 × 28 in Figure 3.21. Instead, in the prediction process, the mask branch is used to generate the mask shape of the object. Figure 3.22 shows the prediction workflow.

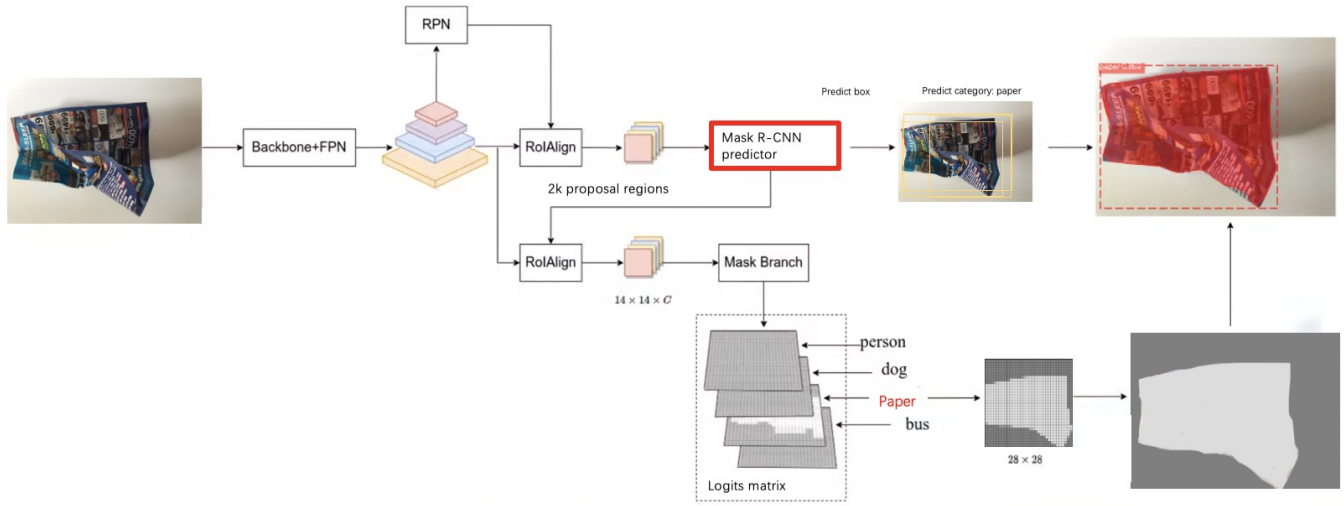

Figure 3.22. Mask branch during the prediction process [29].

Different Mask Shape Size

When training a neural network with high-resolution images, the binary mask representing each ROI will also be very large. For example, if we train an image with a size of 1024 × 1024, the mask corresponding to a single object will need at least 1 MB of memory. If the image contains 10 objects then 10 MB will be needed. Yet most of the values in the mask matrix are equal to 0, which is a waste of space. Instead of storing so many zeros directly, by changing the size of the mask shape the memory needed will reduce. This principle is similar to those used by common compression algorithms. And it is possible to optimize the precision of the mask shape, then save storage space and improve training speed.

To resize the mask shape size, the corresponding convolutional layer needs to be resized too. Taking the 56 × 56 as an example, a ConvTranspose2d function is added at the end of the FEN Layer to enlarge the output size by 2 times. ConvTranspose2d is commonly used to enlarge the size of the network when needed. The definition of convolution is to set the height and width of an image A respectively, and the number of channels. Then use the convolution kernel to do convolution, with the step of stripe, padding, and get B after convolution. Conversely, ConvTranspose2d is to change B back to A. For the ConvTranspose2d operation, the convolution kernel setting is the same as the convolution operation. After inputting feature map B and setting the convolution parameters, feature map A is obtained. This is the step to enlarge the mask shape size. And for a smaller mask shape like 11, another Pooling layer is added to resize the mask shape.

Chapter 4

Test Result and Analysis

In this chapter, the results of forecasting and the evaluation of the models will be shown. Different sizes of the mask shape were tested in order to find the one that could ensure the best performance.

4.1 Model Evaluation

In the Mask R-CNN model, several indexes are used to evaluate the model performance.

4.1.1 Precision And Recall

The results of the object detection process can be divided as follows:

- *True Positive* (TP): for example, a metallic objects are correctly detected as metals.
- *False Positive* (FP): for example, a metal object is misidentified as glass or the background is misidentified as metal.
- *True Negative* (TN): for example, the background is recognized correctly as the background. Note that background identification is out of the scope of interest, ie object detection.
- *False Negative* (FN): for example, a metal object is not detected.

At the end of the test, for a given category, TP and FP are sorted in descending order of probability and different thresholds are taken to obtain the corresponding *Precision* (P) and *Recall* (R). The definition of these indices is shown below:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Then the *Precision-Recall* (PR) curves of all categories can be obtained. The Figures 4.1 - 4.4 shows the PR curves under all categories and with a mask with all size chosen. The area of the PR curve of each category is the average precision (AP) of each category. The area trend is consistent with the data in Table 4.1.

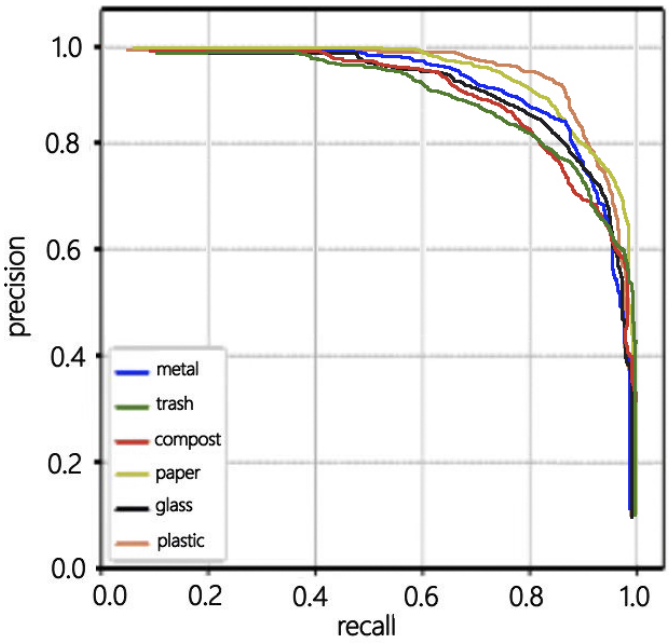

Figure 4.1. PR curve under (11, 11) mask shape size.

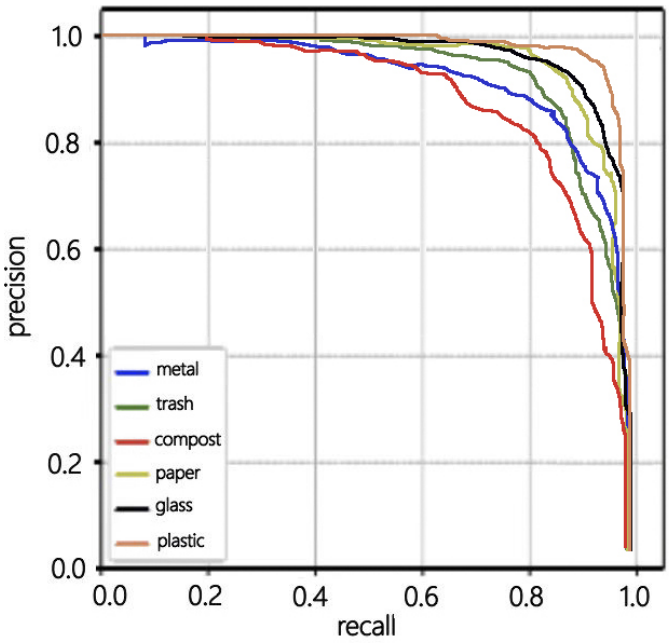

Figure 4.2. PR curve under (28, 28) mask shape size.

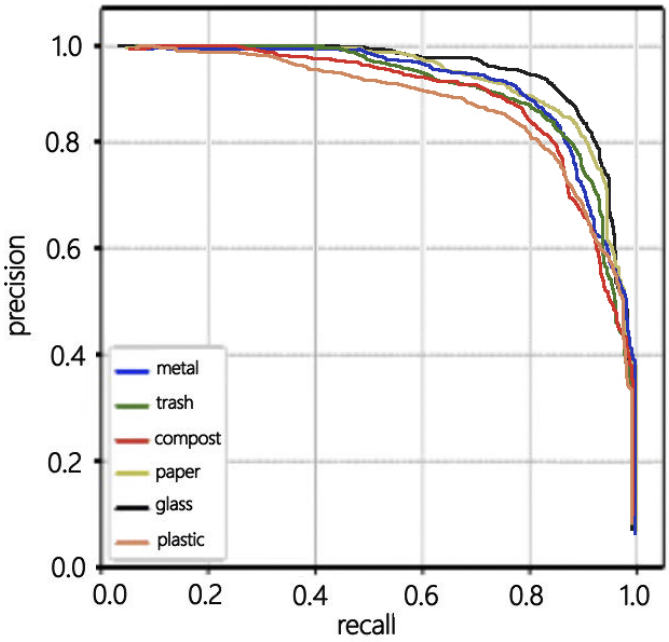

Figure 4.3. PR curve under (56, 56) mask shape size.

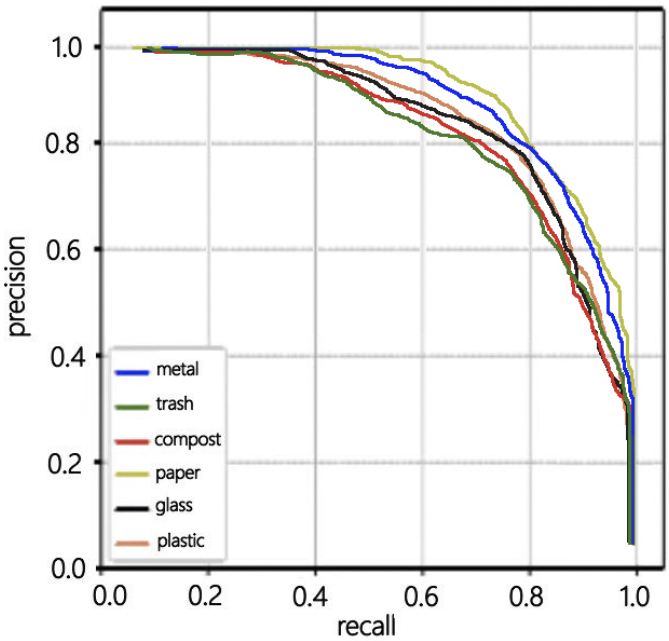

Figure 4.4. PR curve under (112, 112) mask shape size.

4.1.2 AP and mAP

The *Average Precision* (AP) is commonly used in object detection as an evaluation index. Its value corresponds to the area under a certain category of PR curve. The

definition of AP is shown below:

$$AP = \int_0^1 P(R)dR$$

Depending on the threshold of the IoU, it can be divided into variants AP50 and AP75. AP50 is the AP value when the IoU threshold is 0.5, while AP75 is the AP value when the IoU threshold is 0.75. In this thesis, AP50 will be used as an index to evaluate the experimental results, this index can be used to compare the average precision performance of the trained model.

The *Mean Average Precision* (mAP) is the mean value of AP under all categories. The calculation formula of this index is shown below.

$$mAP = \frac{\sum_{i=1}^{k} AP_i}{k}$$

By calculating mAP, it shows the general precision of the Mask R-CNN model. The results of training each network with different mask shape sizes on the training and test datasets are shown in Table 4.1. The average precision of each category on the test dataset and the training dataset is relatively balanced, indicating that the model has learned to distinguish between all problem classes. By observing the mAP of masks of different sizes, it can be seen that the best mask size is the one having 28 x 28 pixels.

Table 4.1. Network AP and mAP on Training dataset (TR) and Test dataset (TE)

Mask Shape Size		AP (IoU = 50)						mAP
		Metal	Plastic	Paper	Glass	Trash	Compost	
(112,112)	*TR*	0.8696	0.88	0.87	0.8696	0.8673	0.8685	0.8692
	TE	0.7811	0.7770	0.7817	0.7728	0.7597	0.7695	0.7736
(56,56)	*TR*	0.8797	***0.89***	***0.89***	0.8796	0.8770	***0.8797***	0.8793
	TE	0.8113	0.7866	0.8135	0.8153	0.8058	**0.7939**	0.8094
(28,28)	*TR*	***0.8851***	0.8817	0.8879	***0.8831***	0.8715	0.8732	***0.8804***
	TE	**0.8139**	**0.8671**	**0.8408**	**0.8554**	**0.8204**	0.7737	**0.8285**
(11,11)	*TR*	0.8776	0.8851	0.8857	0.8799	***0.8773***	0.8695	0.8791
	TE	0.8083	0.8257	0.8187	0.8075	0.7858	0.7934	0.8065

4.1.3 Loss

As introduced in Section 3.2.3 of Chapter 3, the *loss* metric is composed of the classification, bbox, and mask losses.

The Figures 4.5-4.7 separately show the three losses of the trained models with different dimensions of the mask shape. Masks with dimensions greater than 5 x 5 pixels can be considered to have the best performance, as they converge faster and stabilize at a small value. This is because interesting image features are less likely to learn useful information with a too small mask.

This can also be verified by comparing the total loss reported in Figures 4.8 - 4.12. Also in them, the network model having a 112 x 112 mask shape size has the lowest loss. From the total loss, the size $11 > 28 \approx 56 > 112$. The network model with a 112 x 112 mask shape size converges faster under the same conditions. The convergence directly reflects the generalization performance and reflects the precision, but there is no direct proportional relationship when the dataset is small. However, in Table 4.1 it is possible to see that as the size of the mask shape increases, the average precision decreases. This is because, being larger, will create jagged edges for the objects. If these are also close to similar objects and made with the same material, there may be a tendency to select nearby objects. By using a larger mask, the fine details of the image are lost with the consequence of increasing the probability of catching other objects.

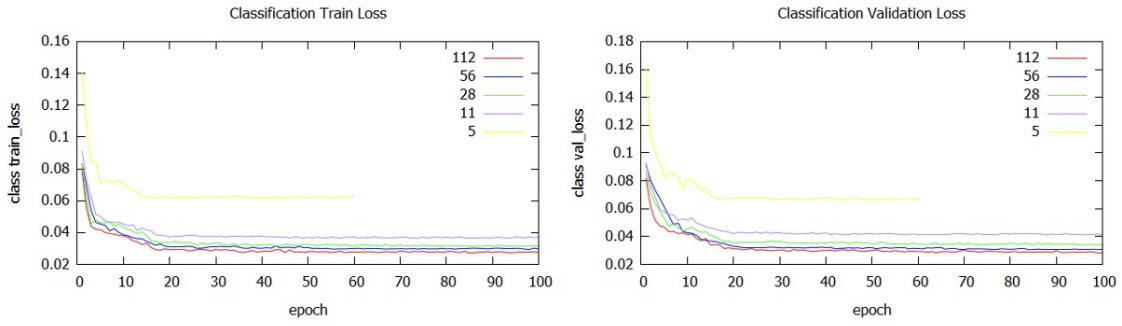

Figure 4.5. Classification loss for different mask sizes.

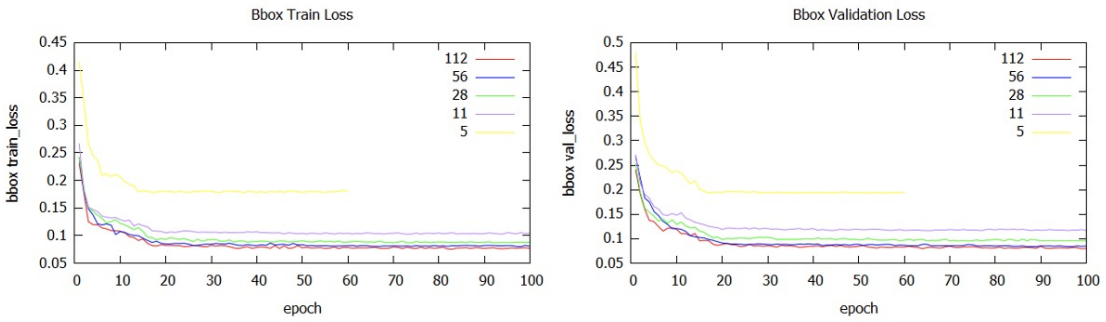

Figure 4.6. Bounding box loss for different mask sizes.

4.2 Prediction Results

Some of the prediction results using the trained network with a mask size of 28 x 28 are shown in Figure 4.13. From the results of the prediction it can be seen that with good light conditions, waste detection is possible. Mask R-CNN has good instance segmentation performance and can correctly distinguish the foreground and background.

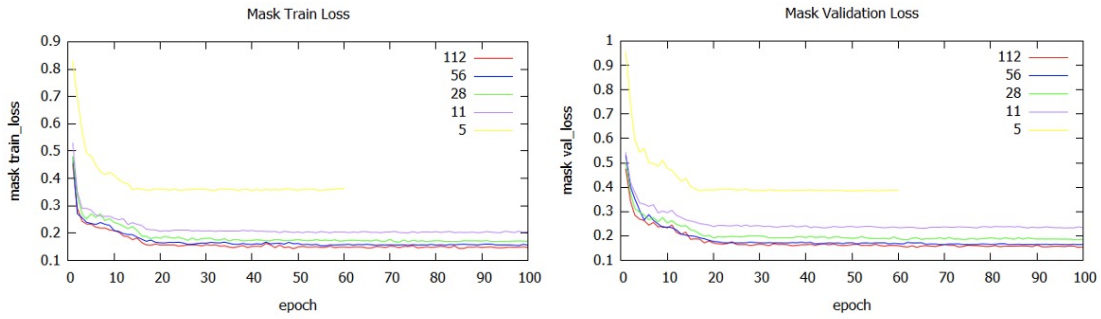

Figure 4.7. Mask loss for different mask sizes.

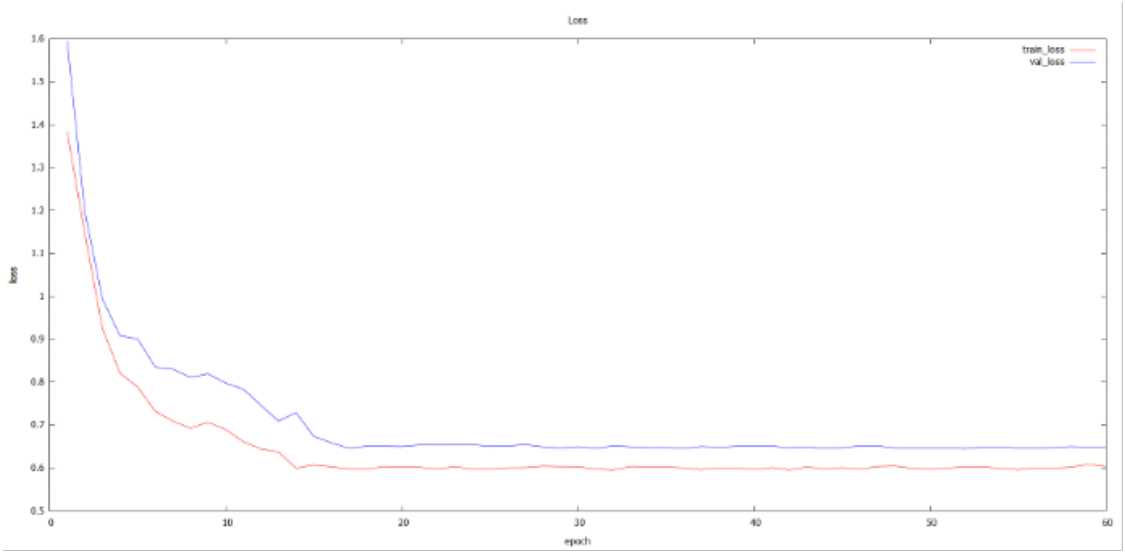

Figure 4.8. Total loss under 5 x 5 mask shape.

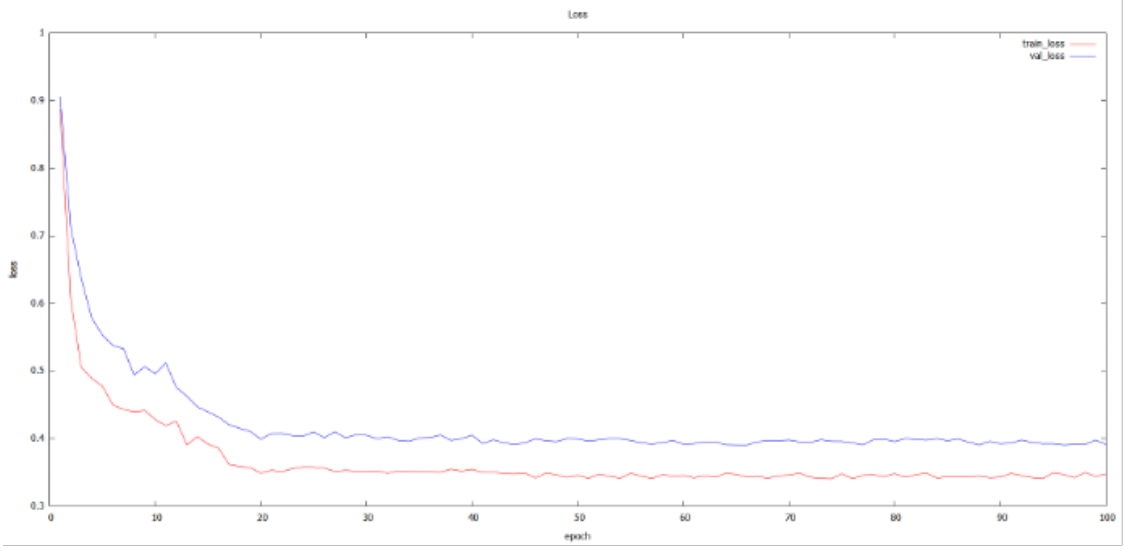

Figure 4.9. Total loss under 11 x 11 mask shape.

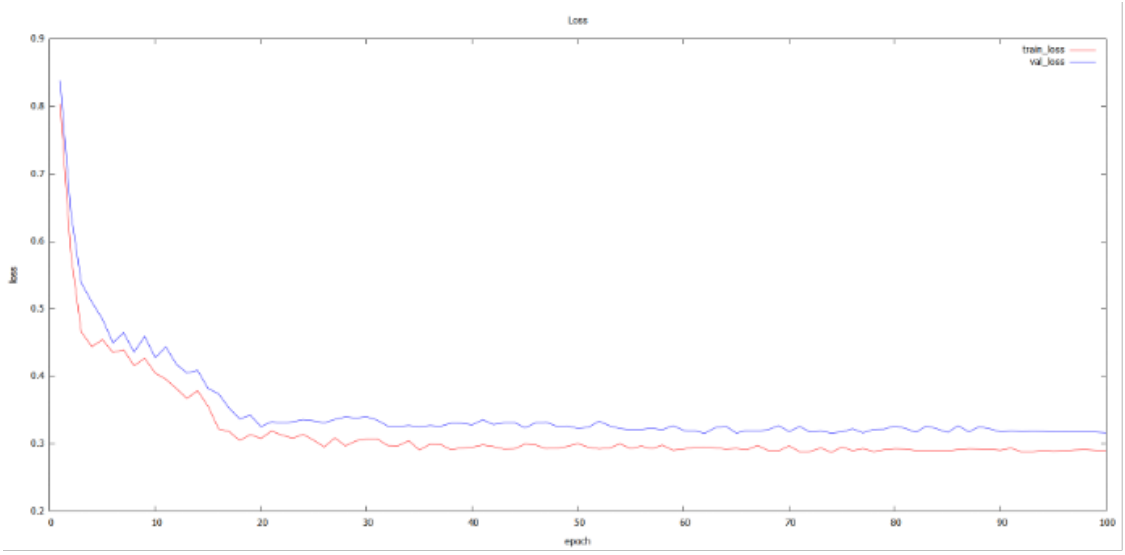

Figure 4.10. Total loss under 28 x 28 mask shape.

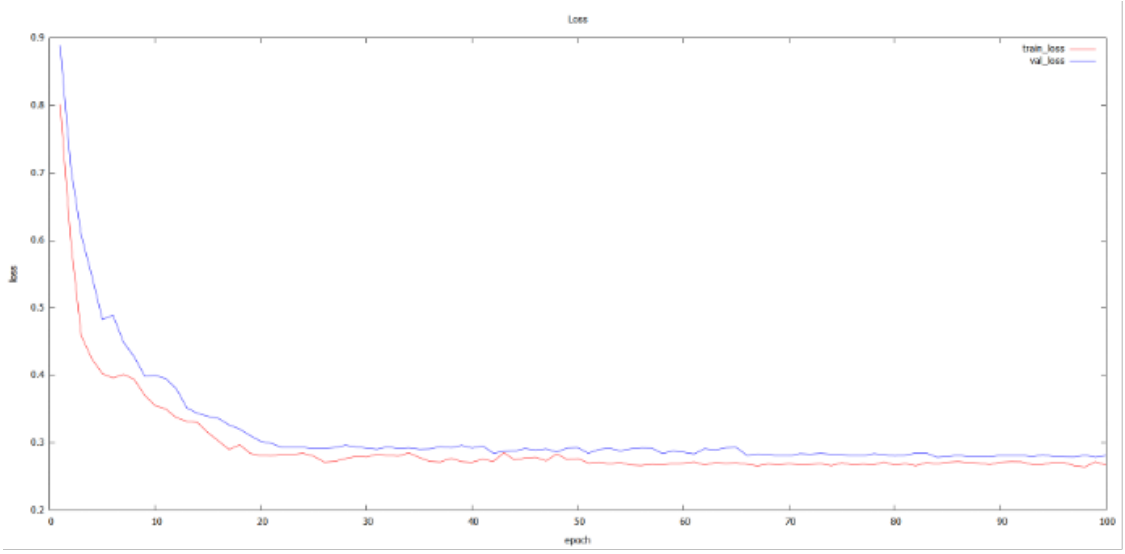

Figure 4.11. Total loss under 56 x 56 mask shape.

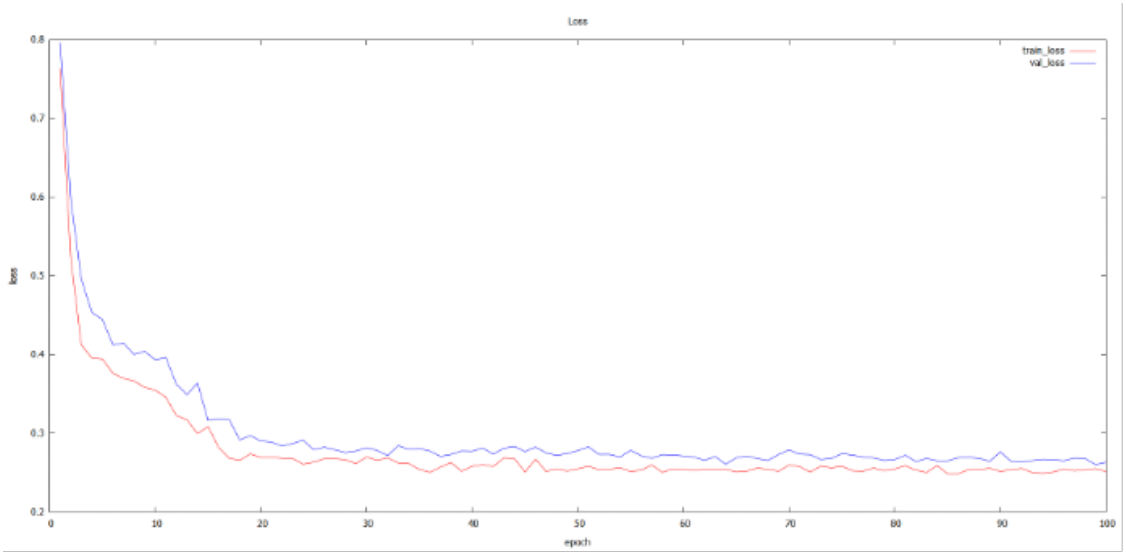

Figure 4.12. Total loss under 112 x 112 mask shape.

Figure 4.13. Example of prediction results [29].

Chapter 5

Conclusion

This final chapter will detail the results achieved and propose some ideas for future improvements to the project.

5.1 Conclusion

Based on the Mask R-CNN algorithm, this thesis completes an object detection model taking trash images as the research object. By training the model on the self-made trash dataset, the instance segmentation of waste objects contained in the image is realized, and the performance of the model is evaluated and analyzed. The experimental results show that the trash object detection model proposed in this paper has good performance. The specific conclusion of this thesis is summarized as follows:

1. A dataset has been created that includes six different types of materials: paper, metal, garbage, compost, plastic, and glass. A label was also assigned for the type of waste, which was not used in the subsequent phase. The total number of different tagged objects in the images provided is 5175.

2. An instance segmentation system based on the R-CNN Mask algorithm has been implemented, capable of distinguishing the aforementioned types of materials. It has been trained on the recycle bin dataset created using different mask sizes. Those were chosen to be 112 x 112, 56 x 56, 28 x 28, 11 x 11, and 5 x 5. In general, the network model with a mask size of 28 x 28 has the best performance, followed by that with a size of 56 x 56. The model having a mask size of 112 x 112 has obvious advantages in convergence speed but a low precision in the test. This is because a mask shape that is too large will create a jagged edge for objects, affecting the precision of the model itself and increasing the tendency to also select similar objects in the immediate vicinity.

5.2 Future Work

Waste recycling is a thorny problem. Although it has been adopted for many years in many countries, there are still challenges in handling garbage properly. New technologies such as deep learning can be a valid help both for this and for other problems that are difficult to solve with traditional methods. Object detection through deep learning techniques allows to perform waste detection with good precision and has the potential to replace manual sorting in the existing separate collection process. It can also be used alongside pre-existing waste detection techniques as a redundant way to improve recycling.

This thesis discussed the problem, the related work on technology, and the implementation of an object detection system based on Mask R-CNN. It can be further improved in several ways:

1. The size of the dataset is quite small. By increasing it, detection accuracy could be improved and it could allow the detection of individual objects, such as paper bags, plastic cups, tin cans, and others. In this regard, separate labeling was created, not used for this thesis but ready for future use.

2. In the test process, easily confused objects were found to have a large impact on actual detection performance (eg paper and plastics are very similar in some perspectives and can sometimes be confusing).

3. No attempt has been made to improve the performance of the algorithm by modifying the structure of the backbone network.

The complexity and systematic nature of the classification of individual waste make it impossible for any new technology to improve its treatment status in a short time. Through a green development path, it is possible to implement the creation of datasets having a sufficient number of samples in various categories. It is also necessary to optimize the algorithms in the field of object detection and run more tests in order to effectively estimate the real practical effect of object detection in waste classification.

Bibliography

[1] Hoornweg, Daniel; Bhada-Tata, Perinaz. 2012. "What a Waste : A Global Review of Solid Waste Management", https://openknowledge.worldbank.org/handle/10986/17388. [Online]. Last accessed: 11/07/2022.

[2] Niehs.nih.gov, "Cancer and the Environment," 46, 2018. https://www.niehs.nih.gov/health/materials/cancer_and_the_environment_508.pdf. [Online]. Last accessed: 11/07/2022.

[3] Ren S., He K., Girshick R., et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2017, 39(6):1137-1149.

[4] Wu Xiang-ning, He Peng, Deng Zhong-gang, Li Jia-qi, Wang Wen, Chen Miao. A deep learning model of small object detection based on attention mechanism[J]. Computer Engineering & Science,2021,43(01):95-104.

[5] Li Cong, Pan Li-li, CHEN Rong-yu,et.al. A novel fusion DCNN for image classification[J]. Computer Engineering & Science, 2019, 41(12):2179-2186.

[6] Krizhevsky A., Sutskever I, Hinton G. Image Net Classification with Deep Convolutional Neural Networks[J]. Advances in neural information processing systems, 2012, 25(2).

[7] Russakovsky O., Deng J., Su H., et al. Image Net Large Scale Visual Recognition Challenge[J]. International Journal of Computer Vision, 2015, 115(3):211-252.

[8] Girshick R., Donahue J., Darrell T., et al. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014.

[9] Girshick R. Fast R-CNN[C]// Proceedings of the IEEE International Conference on Computer Vision. 2015:1440-1448.

[10] Evan Shelhamer, Jonathan Long, Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation[M]. IEEE Computer Society, 2017.

[11] Heaton, Jeff. Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning[J]. Genetic Programming and Evolvable Machines, 2017:1-3.

[12] Kim, Yoon. Convolutional Neural Networks for Sentence Classification[J]. Eprint Arxiv, 2014.

[13] P. A. Viola, M. J. Jones, D. Snow. Detecting pedestrians using patterns of motion and appearance [C].IEEE Conference on ICCV, 2003:734-741.

[14] Navneet Dalal, Bill Triggs. His tograms of oriented gradients for human detection [C]. IEEE Conference on CVPR, 2005, 1: 886-893.

[15] Chen P. H., Lin C. J., Schlkopf B. A Tutorial on v-support vector machines[J]. Applied Stochastic Models in Business and Industry, 2005, 21(2):111-136.

[16] Smola A. J., Schlkopf B. A tutorial on support vector regression[J]. Stats and Computing, 2004,14(3):199-222.
[17] V. D S A. Advanced support vector machines and kernel methods[J]. Neurocomputing, 2003, 55(1/2):p.5-20.
[18] Felzenszwalb P. F., Girshick R. B., McAllester D., et al. Object detection with discriminatively trained part-based models [J]. IEEE transactions on pattern analysis and machine intelligence, 2010, 32(9): 1627-1645.
[19] Goyal P., Dollár, Piotr, Girshick R., et al. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour[J]. 2017.
[20] Redmon J., Divvala S., Gishick R., et al. You only look once: unified, real-time object detection [C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016: 779-788.
[21] Liu W., Anguelov D., Erhan D., et al. SSD: Single Shot MultiBox Detector [C]. European Conference on Computer Vision. 2016: 1408-1421.
[22] Redmon J, Farhadi A. YOLO9000: better, faster, stronger, arXiv preprint, 2017.
[23] Redmon J, Farhadi A. YOLOv3: An Incremental Improvement, arXiv preprint arXiv:1804.02767, 2018.
[24] He K, Gkioxari G, Dollar P, Mask-RCNN[J]. IEEE Transactions on Pattern Analysis&Machine Intelligence, 2017:2980-2988.
[25] Kirillov A., Wu Y., He K., et al. PointRend: Image Segmentation as Rendering[J]. 2019.
[26] Beeren Sahu, https://towardsdatascience.com/the-evolution-of-deeplab-for-semantic-segmentation-95082b025571. [Online]. Last accessed: 11/07/2022.
[27] Everything about Mask R-CNN: A Beginner's Guide. [Online]. Last accessed: 11/07/2022.
[28] ReLearn Official Site, https://www.re-learn.eu/en/. [Online]. Last accessed: 11/07/2022.
[29] Wastedata-Mask_RCNN-multiple-classes, https://github.com/SriRamGovardhanam/wastedata-Mask_RCNN-multiple-classes/tree/master/main. [Online]. Last accessed: 11/07/2022.
[30] Nando Device Introduction, https://www.re-learn.eu/en/reset/. [Online]. Last accessed: 11/07/2022.
[31] VGG Image Annotator (VIA), https://www.robots.ox.ac.uk/~vgg/software/via/. [Online]. Last accessed: 11/07/2022.
[32] The PASCAL Visual Object Classes Homepage, http://host.robots.ox.ac.uk/pascal/VOC/. [Online]. Last accessed: 11/07/2022.
[33] COCO Dataset, https://cocodataset.org/#home. [Online]. Last accessed: 11/07/2022.
[34] Kaiming He, Xiangyu Zhang, Deep Residual Learning for Image Recognition, 2015, https://doi.org/10.48550/arXiv.1512.03385.
[35] Karen Simonyan and Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", 2014, arXiv:1409.1556.