Recognition of Graphological Wartegg Hand-drawings

Yunqi Xu

A Thesis
in
The Department
of
Gina Cody School of Engineering and Computer Science

Presented in Partial Fulfillment of the Requirements
For the Degree of
Master of Computer Science (Computer Science) at
Concordia University
Montréal, Québec, Canada

September 2022

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: **Yunqi Xu**
Entitled: **Recognition of Graphological Wartegg Hand-drawings**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. Charalambos Poullis

_____ Examiner
Dr. Eugene Belilovsky

_____ Examiner
Dr. Charalambos Poullis

_____ Thesis Supervisor
Dr. Ching Y. Suen

Approved by _____
Dr. Leila Kosseim, Graduate Program Director

_____ 2022 _____
Dr. Mourad Debbabi, Dean
Gina Cody School of Engineering and Computer Science

Abstract

Recognition of Graphological Wartegg Hand-drawings

Yunqi Xu

Wartegg Test is a drawing completion task designed to reflect the personal characteristics of the testers. A complete Wartegg Test has eight 4 cm x 4 cm boxes with a printed hint in each of them. The test subjects are required to use a pencil to draw eight pictures in the boxes after they saw these printed hints. In recent years, the trend of utilizing high-speed hardware and deep learning based model for object detection makes it possible to recognize hand-drawn objects from images. However, recognizing them is not an easy task, like other hand-drawn images, the Wartegg Test images are abstract and diverse. Also, Wartegg Test images are multi-object images, the number of objects in one image, their distribution and size are all unpredictable. These factors make the recognition task on Wartegg Test images more difficult. In this thesis, we present a complete framework including PCC (Pearson's Correlation Coefficient) to extract lines and curves, SLIC(Simple linear Iterative Clustering Algorithm) for the selection of key feature points, DBSCAN(Density-based spatial clustering of applications with noise) for object cluster, and finally utilize transfer learning to increase the converging speed during training and deploy YoloV3-SPP model(A deep learning network) for detecting shapes and objects. Our system produced an accuracy of 87.9% for one object detection and 75% for multi-object detection which surpass the previous results by a wide margin.

Acknowledgments

First and foremost, I would like to express my deep and sincere gratitude to my research supervisor, Professor Ching Y. Suen of Concordia University. I am deeply indebted to him for his patience and guidance during the past three years of study. Without his assistance, this thesis would have never been accomplished. Prof. Suen is an experienced guide, a learned professor, and a responsible supervisor. He provided useful suggestions when I am encountering difficulties in my research. I am so grateful that I have learned from him, and these will be my precious wealth in the future.

I am extremely grateful for the financial support from NSERC. Their tangible support makes my study life easier.

I would like to thank all the members of CENPARMI for their wonderful collaboration. They supported me greatly and were always willing to help me. Because of their help, I have a relaxed and comfortable life studying and working here. After this, I would like to say special thanks to Phoebe Chan for her editing and proofreading help with my thesis. Also, I would like to thank my lab-mate Lili Liu for her previous job and research on Wartegg Test. Without her research and workes, I could not start my thesis such quickly.

Most importantly, none of this could have happened without my parents. I want to thank them for their comprehensive and selfless support. Their support and concern have allowed me to pursue my dreams with all my heart. Lastly, I'd like to mention my friends for their emotional support. They always believed in me, encourage me and listened to my complaints patiently, helped me adjust my negative emotions, so that I could have the

courage to face difficulties.

I have had a wonderful experience at Concordia University in the past three years. I believe this experience will become a precious sight in my life journey.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction of Wartegg Test

The Wartegg Test, also called Wartegg Zeichen Test, was developed by Ehrig Wartegg in 1930, which is a classic psychology test that can reflect personalities of a tester [31]. A Wartegg Zeichen Test form is an A4 paper consisting of eight 4 cm x 4 cm squares in two rows with a simple printed sign in each square, as shown in Fig. 1. The testers are required to draw anything in their mind in each square using pencils, also the simple printed sign in each square should be part of the drawing. Recognizing testers' drawings inside those boxes correctly can help graphologists detect the tester's thoughts and predict the tester's potential psychological problem. In our experiment, we followed the above descriptions of Wartegg Test to collect our Wartegg Zeichen Test forms and scanned them into computer, storaged as digital format. Finally split each Wartegg Zeichen Test form into eight independent images [18].

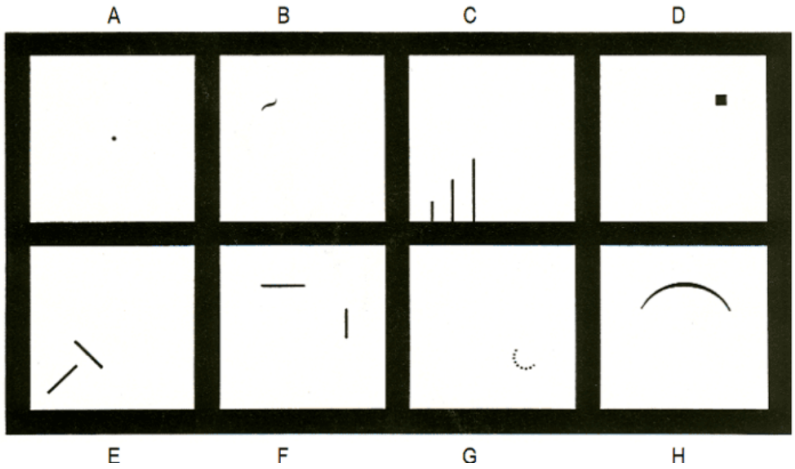


Figure 1: Wartegg Test

In Fig. 1, there are eight squares, and below are some descriptions about these eight squares [2].

- Square A: A point located in the center of the square and the remaining is blank.
- Square B: A short curved line located in the upper left corner of the square.
- Square C: Three gradually longer and parallel straight lines are perpendicular to the bottom boundary of the square also located in the bottom left corner.
- Square D: A small square filled with black colour located in the upper right of the square.
- Square E: Two perpendicular lines composed of a "T" shape without intersection and located in the bottom left of the square.
- Square F: Two straight lines perpendicular to each other without intersection. One is horizontal and another is vertical.
- Square G: Many small dots form a half circle located in the bottom right of the square.

- Square H: A curved radian locates in the upper half of the square.

1.2 Difficulties in Detecting Wartegg Test Images

In recent years, the huge amount of deep learning algorithms for object recognition make it possible to detect those hand-drawn images in Wartegg Test images using a computer. However, many problems become obstacles and make the detection of Wartegg Test image set very difficult. Wartegg Test images are hand-drawn images, so the challenges of recognizing Wartegg Test images are the same as the other hand-drawn images recognition tasks [32]. However, these challenges of recognition of Wartegg Test are not limited to those of hand-drawn images. The difficulties are listed below:

Abstraction and complexity Hand-drawn images are abstract. A complex object in real world can be presented as a simple shape in hand-drawn image.

Lacking colour context information Hand-drawings are black-white images. Their color context is missing compared with the real world colourful images taken by camera.

Diversity Hand-drawn images are diverse because different people have different drawing styles. Same representation may indicate different objects. Also, same object may have different expressions by different testers.

Unpredictability Wartegg Test images are unpredicable, not only the number of objects in each square, but also the size of an object. The signs in these squares will interrupt the thinking of the testers which guide them to draw something surrounding these small signs instead of the center of squares.

Dificiency in training images Many factors influence the performance of the network. First is the number of trianing samples. Usually, to get a detector with high performance, we need to input a large number of images to make sure the network is robust enough. The second is the feature distributions betwwen training and testing samples The more similar the training and testing data distributions are, the higher prediction result a network can achieve. But current open souce hand-drawn image sets, such as QuickDraw [9], Sketchy [25] and Tu-Berlin [4] are one object images with the same size that locate in the center of the images, compared with the Wartegg Test images which comtain multiple objects in a single image. So, directly using these open source images as training images and testing on Wartegg Test images will not get a satisfactory result, and this is another difficulty we need to tackle.

1.3 Contribution

To make it possible to recognize Wartegg Test images, in this thesis, we presented a complete Wartegg Test images framework, that combines the PCC (Pearson correlation coefficient) for the extraction of lines and curves, SLIC (simple linear iterative clustering) + DBSCAN (Density-based spatial clustering of applications with noise) for object segmentation. These methods solve the problem of distribution differences between training images and Wartegg Test images and partly increase the robustness of the neural network. Finally, we used transfer learning to load the pre-trained ImageNet parameters into our YoloV3-SPP backbone network DarkNet53 to increase the converging speed of our network, and then trained a YoloV3-SPP for shape detection.

1.4 Outline of this Thesis

The content of this thesis is organized as following:

- Chapter 1 will introduce the background of Wartegg Zeichen Test, the difficulties when recognizing objects in Wartegg Test images, and the overall contributions of this thesis.

- Chapter 2 provides a brief review of the development history in terms of the area of deep learning in object detection, hand-drawn image sets and the deep learning in hand-drawings.

- Chapter 3 introduces those methods we utilized to process our Wartegg Test Image set. We used PCC for line and curve extraction, SLIC to select key points and DB-SCAN algorithms to segment objects based on these selected keypoints. Finally we utilized the YoloV3-SPP to detect objects from Wartegg Test images.

- Chapter 4 presents our experiment result based on the methods that we introduced in Chapter 3.

- Chapter 5 gives a conclusion of our experiment and discusses the future studies.

Chapter 2

Literature Review

2.1 Review of Object Detection

In 1998, Lecun [15] introduced LeNet-5, which has three Convolutional layers and two pooling layers, achieved 90.56% classification accuracy rate on the handwritten image set MNIST. In 2012, [14] presented AlexNet, which used GPU to learn parameters of the network not only improved the training speed but also achieved a better result. Following this, in 2014, many networks with different structures were presented, such as VGG [26], GoogLeNet [28] and its updateed model Inception V2 [13]. However, simply increasing the depth of a network is not enough. In 2015, ResNet was proposed by [12]. ResNet utilizes the residual block breakthrough the degradation problem that usually appears in a deep network and increases the accuracy significantly. Following this trend, Inception V3 [29], Inception V4 [27], DarkNet19 [22] and DarkNet53 [23] were proposed.

Not only are the classification model growing rapidly, but also are the object detection models.

In 2014, [8] mentioned R-CNN, which used region proposals and CNN significently improved more than 30% accuracy and achieved 53.3% on PASCAL VOC 2012 [6]. After that, Fast R-CNN [7], Faster R-CNN [24] were proposed. Moreover, Yolo [21], SSD [19],

and YoloV2 [22], YoloV3 [23], also YoloV3-SPP [35] were published and achieved some good results on PASCAL VOC and COCO [17].

2.2 Review of hand-drawn images

The beginning of applying classification and recognition deep learning model to hand-drawn images can be retrospected from 2012. In this year, Tu-Berlin [4] presented their hand-drawn image set. After this, in 2016 sketchy images was presented [25] with 125 categories. In 2017, Google published one of the largest sketch image sets QuickDraw [9]. In the year 2018, image sets SPG [16] and SketchSeg [30] were published. Finally, in 2019, our Wartegg Test image set [18] was collected which contains 30 categories and more than 900 images. Accompanied with these image sets, some famous deep learning models were mentioned, such as Sketch-net [34], Deep visual sequential fusion net [10], SketchRNN [9] and Sketchmate [33].

2.3 Purpose of our experiment

Although these models mentioned in Section 2.2 have achieved high performance, they are tested on Tu-Berlin, QuickDraw, or some single object image sets. In 2020, [20] used the original YoloV3 to detect the objects in Wartegg Test images. This was the first time that the modern deep learning approach was applied on the detection task of Wartegg Test. However, they only detected those images that contain a single object. As mentioned in Chapter 1, free hand-drawn images are diverse, like our Wartegg Test images, many images have multiple objects in one image. So, directly deploying deep neural network to detect the Wartegg Test images which contain multiple objects may not be a wise choice, and the results are usually unsatisfactory. It is important to explore how to detect multiple objects using current open source images.

Chapter 3

Method

Some previous image processing methods are effective for specific image sets, but can not deal with Wartegg Test images. Many uncontrollable factors influenced the quality of Wartegg Test image set as mentioned in Section 1. The motivation of our experiment is to unify the data distribution between training and testing images and increase the accuracy of the object detection task.

In this chapter, we will firstly introduce some image processing methods from section 3.1.1 to section 3.1.3. Then, we will introduce YoloV3-SPPP in section 3.2.

3.1 Image processing

This section contains three parts: (1) the PCC for feature selection, (2) SLIC for keypoints selection, and (3) the SCANS for object segmentation.

3.1.1 PCC for feature selection

Donati [3] used PCC to extract the features of sketch images based on the relevance of the pixels in an image. By using this method, those pixels neither belong to lines nor curves can be deleted.

Firstly, choose the different kernel sizes to match the line width invariance. the w_{min} represents the minimum line width, and w_{max} as the max line width.

$$\begin{cases} \sigma_0 = w_{min}/b, & (1) \\ \sigma_i = C * \sigma_{i-1}, & (2) \end{cases}$$

In Eq.1 and Eq.2, C is a constant number, and we can find $\sigma_1 = C^1 * \sigma_0$, $\sigma_2 = C^2 * \sigma_0$, $\sigma_3 = C^3 * \sigma_0$ ... and then we can get Eq. 3.

$$\sigma_i = C^i * \sigma_0 \tag{3}$$

Then we will utilize Eq. 3 to generate N kernel, the number of $N = \log_C(w_{max}/w_{min})$, $i \in [0, N-1]$,

to calculate Eq. 4

$$s_i = next\_odd(7 * \sigma_i) \tag{4}$$

Then we will use the Eq. 4 to generate N 2D kernels.

$$kernel_i = Gaussiankernel(si, \sigma) \tag{5}$$

Then we will use the PCC equation to filter each image.

$$PCC_{xy} = \frac{\sum_{j,k}(I_{xy}(j,k) - Avg(I_{xy}))(k_i(j,k) - Avg(k_i))}{\sqrt{\sum_{j,k}(I_{xy}(j,k) - Avg(I_{xy})^2 \sum_{j,k}(k_i(j,k) - Avg(k_i))^2}} \tag{6}$$

In this Eq. 6, (j, k) is each coordinate in I_{xy}, and the I_{xy} will be calculated by using the Eq. 7 indicated below:

$$I_{xy} = Image[x - int(kernel_i) : x + int(kernel_i)][y - int(kernel_i) : y + int(kernel_i)] \tag{7}$$

Finally, we will generate N images by inputtng different $kernel_i$.

$$MPCC = \begin{cases} maxPCC_{xy} & , \mid maxPCC_{xy} \mid > \mid minPCC_{xy} \mid \\ minPCC_{xy} & , otherwise \end{cases} \tag{8}$$

Then we will use Eq. 8 to merge these PCC images.

In the end, we generate a high-pass filter image for the original image and minus the median filter image based on the PCC result image to get our final result.

3.1.2 SLIC Algorithm for superpixel split

SLIC was mentioned by R.Achanta [1], which splits one image into many sub patches (also called superpixels) based on their values and locations. We utilized this algorithm to split an image into many small patches that help us segment objects in our experiment.

The SLIC algorithm is presented in algorithm 1:

Algorithm 1 SLIC algorithm

Input: k; m; $iterations$; $image\_width$; $image\_height$

Calculate $s = \sqrt{(image\_width * image\_height)/k}$;

$centerset = C_k = (l_k, x_k, y_k)_{k=1...k}$

Update each $c_k.x_k$, $c_k.y_k$ to the lowest gradient in $[c_k.x_k - 1 : c_k.x_k + 1, c_k.y_k - 1 : c_k.y_k + 1]$

Set each pixel $p_i.cluster = None$, $p_i.distance = +\infty$, $i = 1$

if i $\leq$ iteration **then**

  for each cluster c_k in C_k **do**

    for each point p_i in $[c_k.x - s : c_k.x + s$; $c_k.y - s : c_k.y + s]$ **do**

      $D = \sqrt{(\sqrt{(p_i.l - c_k.l)^2}/m)^2 + (\sqrt{(p_i.x - c_k.x)^2 + (p_i.y - c_k.y)^2}/s)^2}$

      if D $< p_i.distance$ **then**

        $p_i.distance$ = D

        $p_i.cluster$ = c_k

      end if

    end for

    Update $c_k.x_k$, $c_k.y_k$

  end for

  $i = i + 1$;

end if

In algorithm 1, the input parameters include: k as the number of pitches, m as a constant, the iteration times, image width, and image height. During the calculation process, firstly, we will calculate S which is the area of each pitch. Secondly, we will generate a center set, which contains k center points and each center point stores the information including the coordinates inforamtion, and those points belonging to this center point. l_k represent the light value of the CIELAB space of the image, since our image is a binary

image, so only the light information will be calculated compare with a color image usually have three values of CIELAB color space, l, a and b.

Then, we will update the location of these center points to the lowest gradient point in its area. In each iteration, let each pixel to its nearest center, recalculate the cluster center until we finished the iteration the get the final result.

3.1.3 DBSCAN For Object Segmentation

DBSCAN was mentioned by M.Ester [5], which was designed to discover clusters with arbitrary shapes based on the density of the dataset. The main idea behind utilize DBSCAN comes from the fact that the density inside a cluster is usually higher than the density outside a cluster, and different clusters may have different densities. In our Wartegg Test images, the density of a line or a curve is usually higher in an object, as illustrated in Fig. 2.

(a) Umbrella (b) Rabbit (c) Car

Figure 2: Examples of Density distribution of Wartegg Test images

We used Eq. 9 and calculated the ratio in a 5*5 slide window.

$$densityratio = Count(pixelvalue \neq 0)/(slizewindowsize^2) \tag{9}$$

It is obvious to observe that some areas in Fig. 2 are close to white, and others are black. White means the density score is high, red indicates average. The darker the area is the lower density score the area has, black means the density score is approximately zero.

Those density values near the crossing lines and curves are usually higher than points like Fig. 2(a), or some lines or curves that are not the main parts of an object as seen in Fig. 2(b). Also, the different objects have different density distributions in Fig. 2(c). These images agree with our conjecture, so we utilized DBSCAN to cluster objects in our experiment.

The algorithm of DBSCAN is shown below, in algorithm 2, Eps is the radius length, minpts indicates the minimum number of points.

Algorithm 2 DBSCAN algorithm

Input: Eps, $Minpts$, $Dataset$

Initialization: $clusterid = 0$; $\forall$ point p_i, $p_i \in$ dataset, p_i.cluster = None

for each point p_i in dataset **do**
  if p_i.cluster is None **then**
    clusterset = [$\exists\ p_j$, if dist(p_i, p_j) $\leq$ Eps]
    if clusterset.length $<$ Minpts **then**
      p_i.cluster is Noise
    else
      for each point p_c in clusterset **do**
        p_c.cluster = clusterid
      end for
      Remove p_i from clusterset
      while clusterset is not Empty **do**
        p_{new} = pop(clusterset[0]);$new\_clusterset$ = [$\exists\ P^{'}$, if dist(p_{new}, $p^{'}$) $\leq$ Eps]
        if $new\_clusterset$.length $\geq$ Minpts **then**
          for each point $p^{'}{}_{new}$ in $new\_clisterset$ **do**
            if $p^{'}{}_{new}$.cluster is None **then**
              clusterset.append($p^{'}{}_{new}$); $p^{'}{}_{new}$.cluster = clusterid
            end if
            if $p^{'}{}_{new}$.cluster is Noise **then**
              $p^{'}{}_{new}$.cluster = clusterid
            end if
          end for
        end if
      end while
    end if
  end if
  clusterid += 1
end for

3.2 YoloV3-SPP

The model that we chose in our experiment for the object detection task was YoloV3-SPP [35]. Yolov3-SPP is the third version of the Yolo model with an SPP structure [11] that can improve the precision of the original YoloV3. Fig. 5 presents the overall structure of YoloV3-SPP. The YoloV3-SPP contains two main parts, the DarkNet53 [23] and the followed SPP structure plus some convolutional layers. The former part acts as the backbone of the network and selects features from the input images. The latter detects objects based on the selected features.

3.2.1 DarkNet53

The DarkNet53 was mentioned in 2018 [23]. It is an extension model from DarkNet19 [22], as a classification network that undertakes the rule of feature selection of the input images. The structure of DarkNet53 has been presented in Fig. 3. The DarkNet53 consists of two convolutional layers followed by four residual blocks. Each residual block has 1, 2, 8, 8, and 4 residual layers separately. The residual layer drives from ResNet [12]. Fig. 4 presents the detailed structure of the residual layer. A convolutional layer is inserted between each residual block.

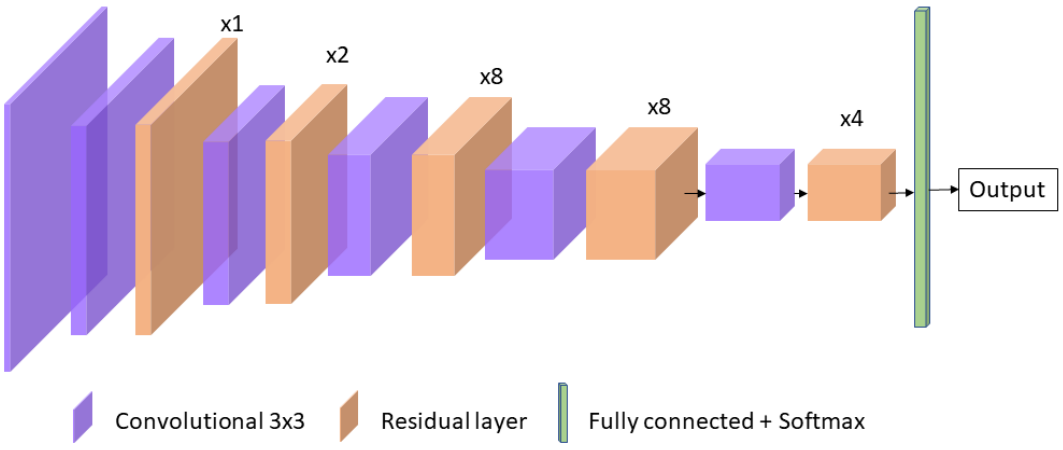


Figure 3: The structure of DarkNet53

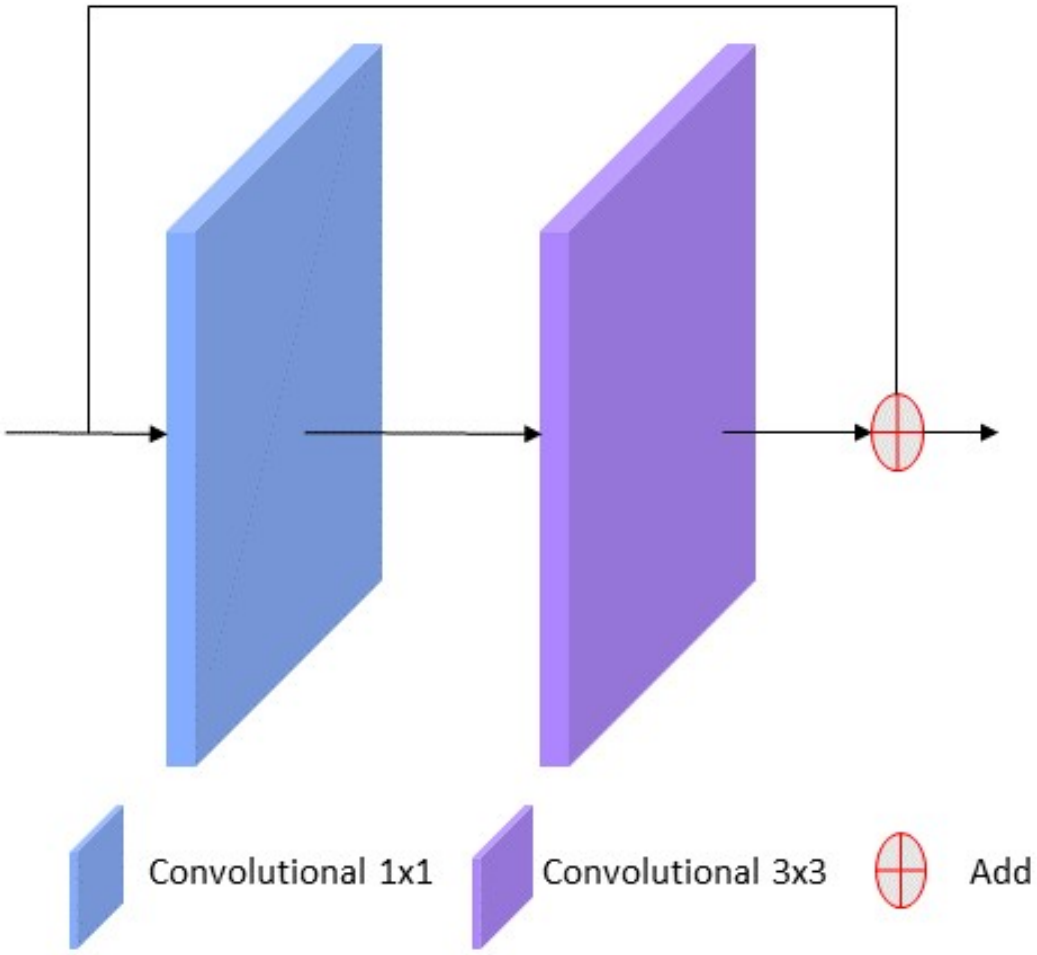


Figure 4: The structure of Residual layer

3.2.2 Structure of YoloV3-SPP

Fig. 5 presents the overall structure of the YoloV3-SPP. Following the DarkNet53, there is an SPP structure [11] with three output branches. Compared with the first output which directly follows the SPP structure and the Convolutional set (the first branch in Fig. 5), the second output and the last output come from the concatenated (the green ball) result of a Residual x 8 block and an Upsample layer (the red box) accompanied by a Convolutional set block.

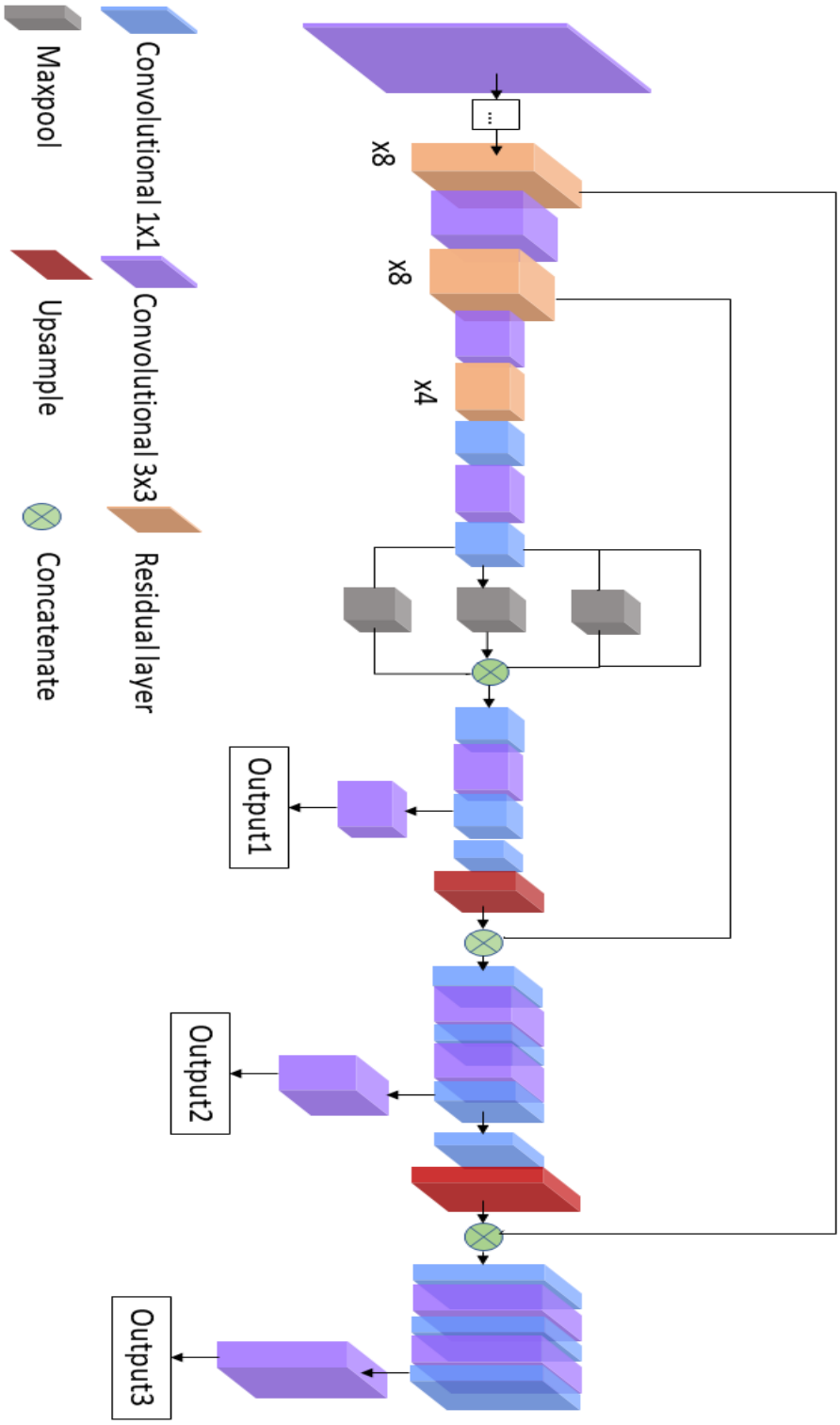


Figure 5: The structure of YoloV3-SPP

Chapter 4

Experiment

Our experiment followed the previous methods introduced in Section 3 to process our Wartegg Test image set. We will firstly introduce our training and test image sets in Section 4.1, and then discuss the pixel value augment in Section 4.2, describe the details of image processing result in Section 4.3, and finally analyze the object detection result in Section 4.4.

4.1 Introduction of dataset

During our experiment, we used QuickDraw [9] as the training image set and our collected Wartegg Test image set as the testing samples that were inputted to validate the performance of the YoloV3-SPP model.

4.1.1 Testing image set

Our Wartegg Zeichen Test form was collected by Liu [18] in 2019. Firstly, we labeled all objects in each image, and then, merged those categories that share the same pattern. For example, face, circle, cookie, tyre and apple are circular patterns, they can be treated as the shape circle. Also, parachutes, hot air balloons, and umbrellas all have a curved ceiling at

the top and a converging tail at the bottom.

Fig. 6 shows the ratio between the area of an object and its image. For those images that have only one object, the area ratio is close to a normal distribution, the median value is around 0.4. On the other hand, for those images that have two or multiple objects, there are more tiny objects in each image.

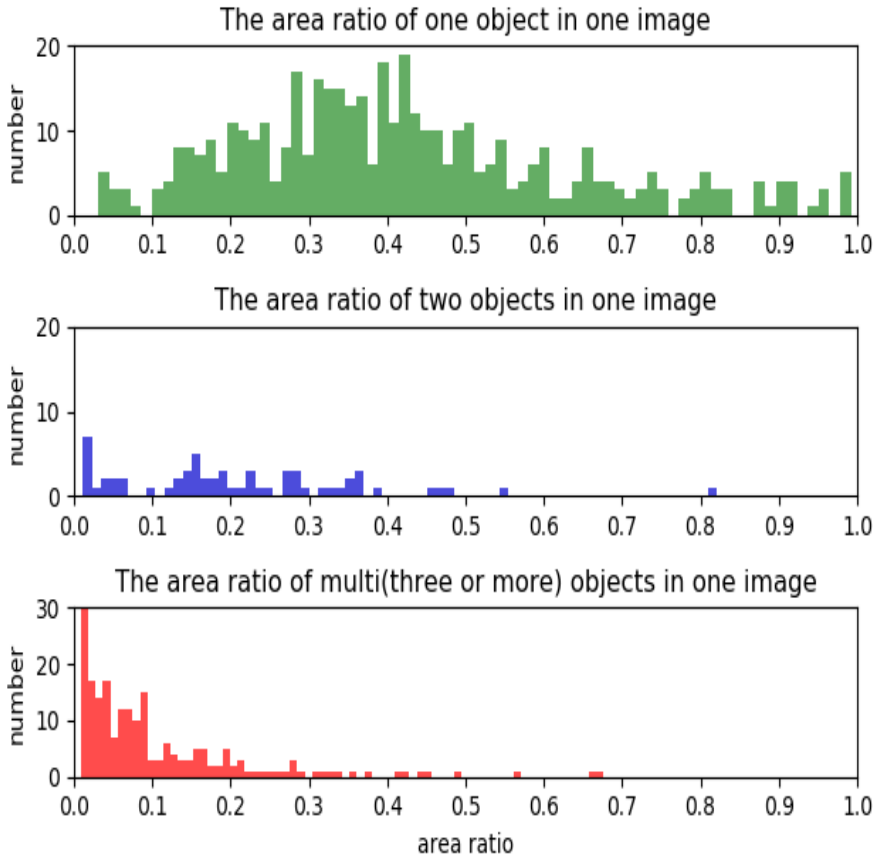


Figure 6: Area Ratio

After labeling and merging, we counted the frequency of each category. Fig. 7 shows those categories that appeared in the Wartegg Test image set. In Fig. 7, those images that contain only one object include more categories. But those images that contain two or multiple objects, the categories concentrate on several categories.

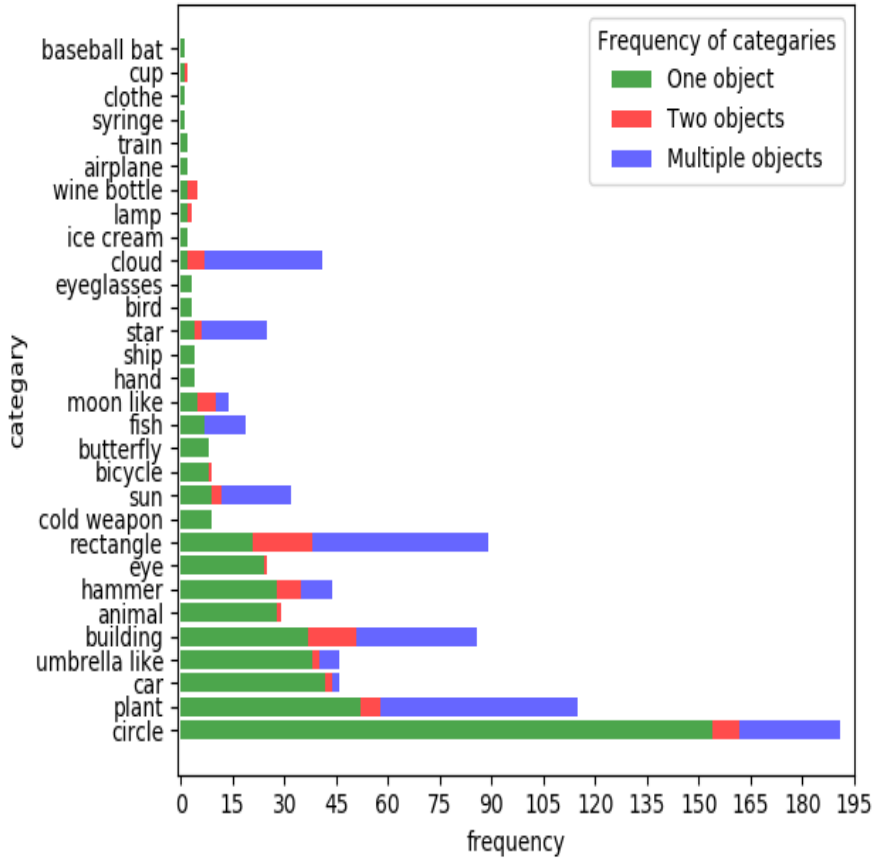


Figure 7: Frequency of categories

Based on previous analysis, we selected the top 11 classes that frequently appeared in those images that contain only one object. The 11 classes are circle, plant, car, umbrella, building, animal, hammer, eye, rectangle, cold weapon, and sun. We cleaned our image set and kept those images containing these categories. After cleaning, we have 439 images with a single object, 55 images have two objects and 93 images contain multiple objects.

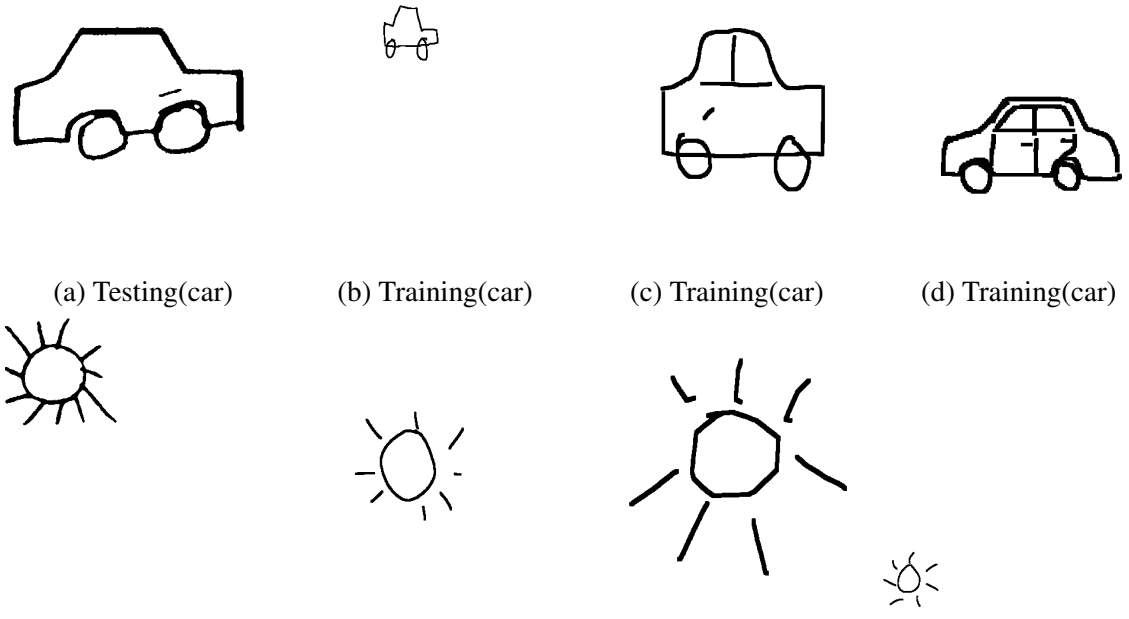
4.1.2 QuickDraw image set

The QuickDraw image set [9] was collected through the game “QuickDraw" developed by Google. The rules of this game are very similar to the famous drawing and guesses game “Darwize“, one person can draw something on paper or screen, and another person guess

what this person drawed. But in the game "QuickDraw", a person draws something on screen, and the machine guesses what it is. The QuickDraw image set contains more than 300 categories, which is suitable for us to select useful categories for training purpose.

We generated the training image set based on the area distribution of the Wartegg Test image set. There are 12000 images in our training image set, each image contains one object and we keep the area ratio of objects with the area of the image from 0.02 to 0.9 to cover most cases in the Wartegg Test image set. Also, the object can be randomly located in the image. The QuickDraw was drawn using a computer mouse or touchpad, or electronic pencil, the width of its lines and curves is only one pixel. But the lines and curves of the Wartegg Test image set are drawn by pencil, so their width varies from 4 to 8 pixels. To balance this difference, we enlarged the width of lines and curves in QuickDraw to 6 pixels.

Fig. 8 shows how we generate training images with different scales, bold lines and curves of the training image set. To do this, the similarities between training and testing images could be maximum.



(a) Testing(car) (b) Training(car) (c) Training(car) (d) Training(car)

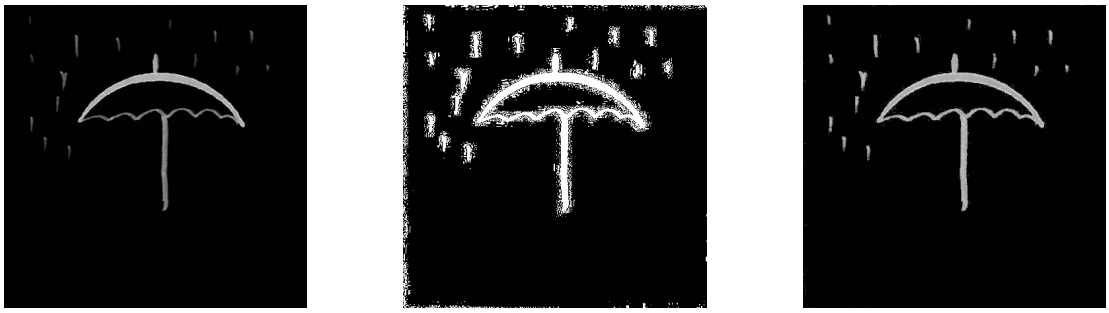
(e) Testing(Sun) (f) Training(Sun) (g) Training(Sun) (h) Training(Sun)

Figure 8: Examples of testing and training images

4.2 Pixel Value Augment

The main purpose of Pixel Value Augment is to enlarge useful information and pretend such information has been deleted by applying PCC and following SLIC + DBSCAN algorithms.

As Fig. 9 shows, although the original Fig. 9(a) looks good, there are lots of noises in the image surrounding the lines and curves, shown in Fig. 9(b).



(a) Original image (b) Actual (c) After augment

Figure 9: Actual Image with noise

By analyzing the Wartegg Test image set, the distribution of a line usually obeys the normal distribution. So for each line, if its width is greater than a width_threshold, we will enlarge its value, by normalization based on its smallest and highest values, in our experiment, the width_threshold is 4.

4.3 Image processing result

Before starting the processing steps, we normalized the value of each pixel in a input image between (0,1), and then followed the steps described in Section 3.1.1 to extract lines and curves.

Fig. 10(a) presents each filter step in the PCC process during our experiment. Fig. 10(b) shows the result after the PCC algorithm. Finally, Fig. 10(c) shows the denoised result. Compared with Fig. 9, those noises surrounding the lines and curves with unpredictable

values are deleted.

(a) Pcc 3 (b) Final result of pcc (c) Denoised result

Figure 10: PCC result

Not only the salt and pepper noise pixels are noises, but during our detection process, those pixels with useless information (drops of rain in Fig. 11) also need to be treated as noise since this will impact our final inference result. So, we firstly split whole image into superpixels. Those red points on Fig. 11 are the centers of the superpixels.

Figure 11: Superpixel result of Umbrella

Once we got the centers of superpixels, we kept those located on lines and curves as the key points of each object and used DBSCAN mentioned in Section 3.1.3 to classify these

key points into multiple clusters.

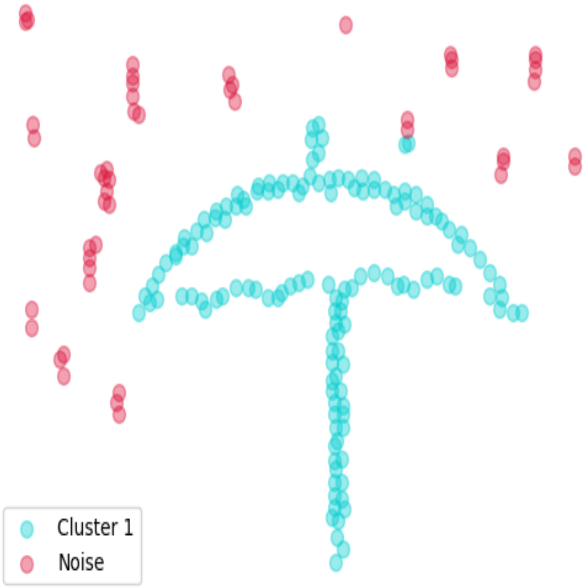


Figure 12: DBSCAN result of Umbrella

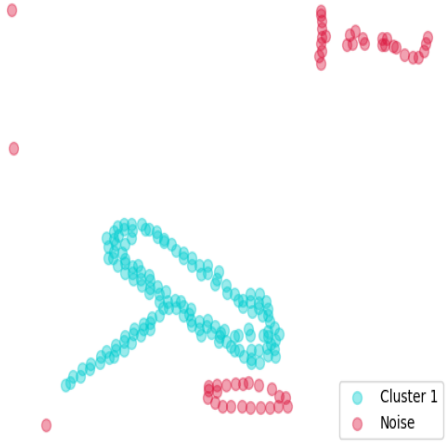
In Fig. 12, it is easy to find that the drops of rain are more sparse when compared with the umbrella, in which most center points located on the umbrella are dense and connected. Based on these features, the DBSCAN algorithm can extract the object and remove the noisy points.

The final result is illustrated in Fig. 13, based on the clustering result after applying the DBSCAN algorithm, we can find the boundary of an object.

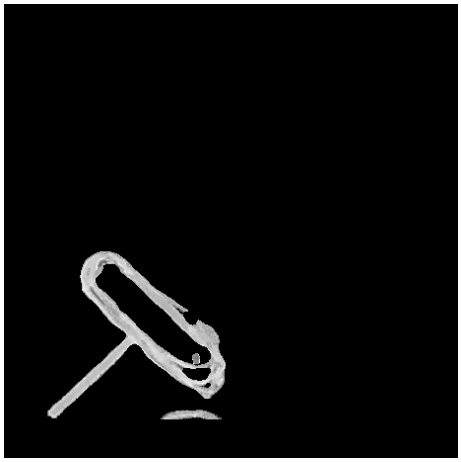
Figure 13: Split result of umbrella

There are also some other examples to indicate the availability of our method, as seen in the Figs. 14, 15, 16, 17.

(a) One object



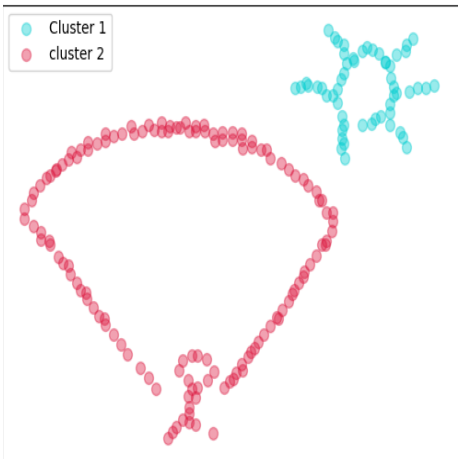
(b) One object



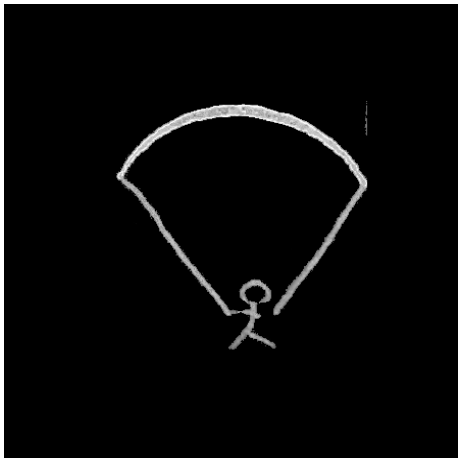
(c) Hammer

Figure 14: Processing result of hammer

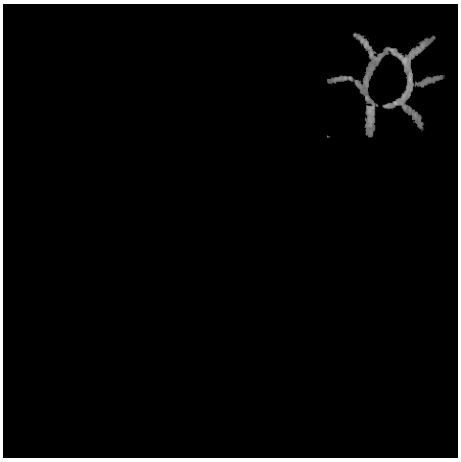
(a) two objects



(b) two objects



(c) parachute

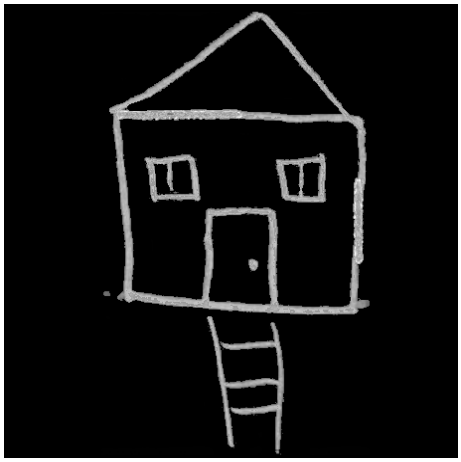


(d) sun

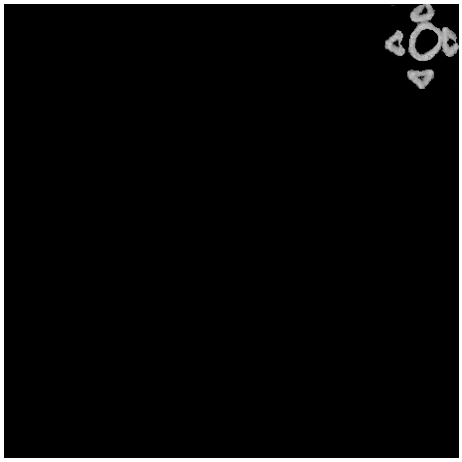
Figure 15: Processing result of Parachute

(a) three objects

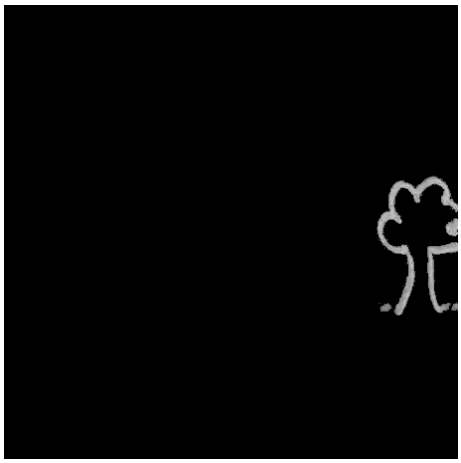
(b) three objects



(c) house



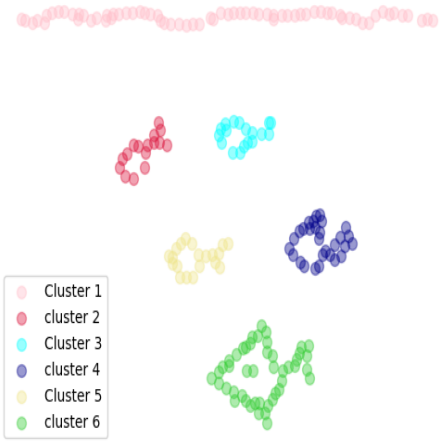
(d) sun



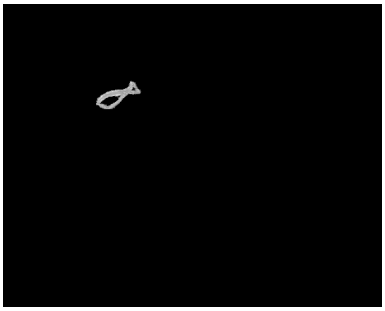
(e) tree

Figure 16: Processing result of House

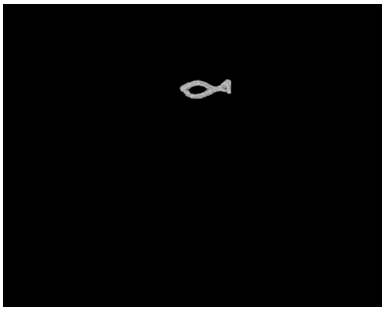
(a) multi objects



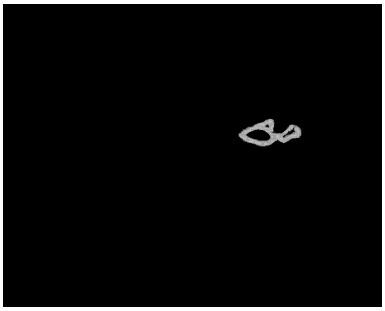
(b) multi objects



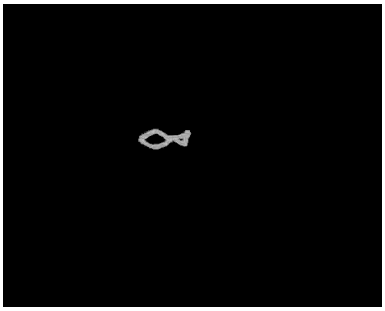
(c) fish



(d) fish



(e) fish



(f) fish

(g) fish

Figure 17: Processing result of Fish

4.4 YoloV3-SPP Object Detection

Note that the contour distributions of natural images and hand-drawn images are similar. In the training process, we use transfer learning-based approach that first loads the parameters of pre-trained DarkNet53 on ImageNet and fine-tuned the last layer using the original QuickDraw images to make the backbone network learn the features of sketch images. Transfer learning can help us solve the problem of the insufficient number of training images and increase the converging speed. In our experiment, the pre-trained parameters from well-trained DarkNet53 on the ImageNet are loaded, then the last layer are fine-tuned to save time, usually 5 epochs to converge. Compared with directly training a DarkNet53 on QuickDraw images, which needs around 15 epochs to let the network coverage, and around 20 epochs to let the result of the network to be come stable. Once the training process of DarkNet53 is completed, we loaded these parameters without the last fully Connected Layer of DarkNet53 into the YoloV3-SPP Network. Using our generated QuickDraw image set, we trained the YoloV3-SPP for the detection task.

During the testing process, we have three image sets, Image set 1 contains those images that originally have only one object, Image set 2 contains those images which originally have one or two objects, and Image set 3 contains those images that the original images have one, two or multiple objects. We inputted each split Wartegg Test image into the trained YoloV3-SPP model to get the inference result. We got the inference results for each image, then, we will merge those split images which belong to a same image together with their predicted bounding boxes. Finally, we can calculate the performance of the YoloV3-SPP model. We will use the COCO image criterion to evaluate our result. The testing result is shown in Table 1.

Table 1 shows that we got a much better result compared with the previous result, which got 68.72% (IoU = 0.5) for one object detection [20], and we increased this score to 87.9% (IoU = 0.5) with an mAP of 67.7%. However, the result drops when testing on one and two

Table 1: The result of YoloV3-SPP on Wartegg Test Image set

Method	Image set	AP_{50}	AP_{75}	mAP
Ours	1	87.9%	76.3%	67.7%
	2	78.9%	66.8%	56.7%
	3	75.0%	57.1%	51.8%
Without ours	(original)1	81.2%	73.8%	62.5%
	(original)2	57.8%	56.2%	51.4%
	(original)3	52.6%	46.4%	43.9%

objects, 78.9% for Iou= 0.5, and the mAP is 56.7%. The performance drops continuously for the detection task of one, two, and multiple objects, i.e. 75.0% when IoU=0.5 and 51.8% for mAP Score. By analyzing this phenomenon, the main reason is that, with an increase in the object number, the area ratio of each object with its image size will decrease. In comparison, for multiple object detection, there are more tiny objects in the testing image set compared with one object image set. As Fig. 6 shows, these tiny objects will lower the performance of the network. On the other hand, as a comparison, we also use the original images as the testing image set to verify the performance of the trained detector on the original Wartegg Test image set. The result has decreased significantly, especially for image sets 2 and 3.

4.5 Analysis of Result

By analyzing the detection result of the network with the main error still comes from the difference of pattern from training and testing images. And it is a common sense that a network will only learn those inputted into it. There are some aspect of difference we can consider as a problem which influence the performance of the network.

1. Split can not remove everything completely.

During our split process, the DBSCAN cluster is based on the center points of super-pixels. Then, we split each object based on the approximate boundary instead of the precise outline. So, the split result may contain part of other objects, and this will influence the detection accuracy of the network, as shown in Fig. 18(a)

2. Similar patterns

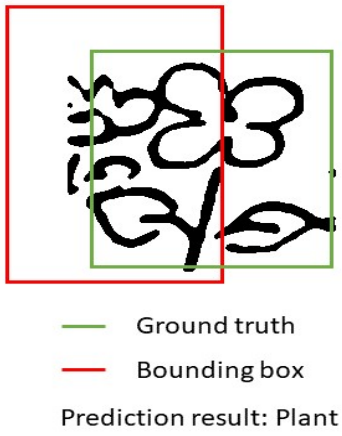
   In our images, some objects may contain a pattern that is very samilar to other category. As shown in Fig. 18(b), part of the eye is similar to the sun. It is true that in our training image set, some suns look like the pattern in the red bounding box. This kind of pattern is the main feature of the sun. The similar patterns from different categories may also influence the performance of our model.

3. Extremely abstract object

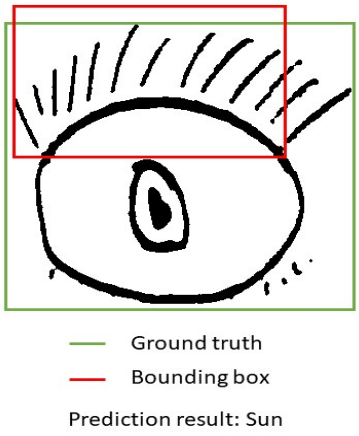
   The objects drawn from our subjects are truely more abstract than what we have in our traininig images. Some subjects use many dense dots to draw something. From the view of human, these objects can be detected without hesitation, but difficult for machine. Fig. 18(c) and Fig. 18(d) present some abstract images in our testing image set.

4. Insensitive to tiny objects

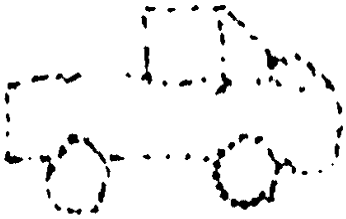
   As we all know, the more objects in an image, the more tiny they are. However, networks are insensitive to these too tiny objects. Some images are missed by the detection process leading to a low accuracy rate.



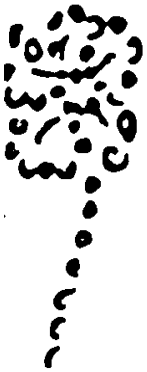
(a) Error 1



(b) Error 2



(c) Error 3



(d) Error 4

Figure 18: Examples of errors

Chapter 5

Conclusion and future work

5.1 Conclusion

In this thesis, we present a complete framework of the Wartegg Test image set in object detection. Our method becomes a bridge between a single object hand-drawn image set and multiple object detection task. Our method does not only simplify the multi-object detection task of Wartegg Test image set, but also improves the result of the performance of Wartegg Test image set in object detection. However, there are still some limitations in our approach. Firstly, the DBSCAN cluster is not suitable for those objects tightly connected or overlapped together, the DBSCAN will treat them as one object which will not be recognized by YoloV3-SPP. Secondly, the parameters of DBSCAN are manually modified, although we have already classified images based on their object area and batch processing the images, we still need to modify the parameters several times. Finally, YoloV3-SPP may not be suitable for tiny object detection, the next step may need to design part of the network which is suitable for both large and tiny objects to improve the performance.

5.2 Future Work

For future work, there are some different valuable research directions, such as color generation for a sketch image, finding the skeleton of these hand-drawn pencil sketches, transforming these pixel format images into vector format images and finding new method to improve the segmentation method. Many of these directions still lack research that are worth paying attention to.

The intellengence of segment method can be improved

Some limitations of our current segment method are obvious. In the next step, we can improve the segment processing to split the object based on their outline.

Add more categories

In our experiment, we merged together some of the categories which have the same shape. But an intellgent model should detect more patterns based on details. So, our next step may focus on designing a new model that can detect more categories.

Focus on utilizing small image set

In our experiment, we used a huge amount of images from published open-source image sets to achieve a good accuray and trying to make our network robust. However, as we all know, humans have the ability of transform knowledge. For example, in our life, we only learnt how to draw one kind of car and then we can draw many different types of cars, most of them have the same features, such as four tyres. But a machine needs to learn all different cars. So, for the next step, we may focus on how to establish a model that can trasfer more information from the real world and reduce the reliability of using huge training images.

Some creative works

There are also some other works we can do in the future. In recent years, many works focus on generating stylish real world images or face images. For us, may be we can also utilize Generative Network to stylish our images, colorize images, and let machine describe an image in several words or sentences.

Publication

Yunqi Xu and Ching Y. Suen, "Recognition of Graphological Wartegg Hand-drawings", in press, Proc. The 20th Conference of the International Graphonomics Society. Spain, 2022.

Bibliography

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

[2] A. Crisi and F. Dentale. The wartegg drawing completion test: Inter-rater agreement and criterion validity of three new scoring categories. *International Journal of Psychology and Psychological Therapy*, 16(1):83–90, 2016.

[3] L. Donati, S. Cesano, and A. Prati. A complete hand-drawn sketch vectorization framework. *Multimedia Tools and Applications*, 78(14):19083–19113, 2019.

[4] M. Eitz, K. Hildebrand, T. Boubekeur, and M. Alexa. Sketch-based image retrieval: Benchmark and bag-of-features descriptors. *IEEE Transactions on Visualization and Computer Graphics*, 17(11):1624–1636, 2011.

[5] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[6] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[7] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[9] D. Ha and D. Eck. A neural representation of sketch drawings. *arXiv preprint arXiv:1704.03477*, 2017.

[10] J.-Y. He, X. Wu, Y.-G. Jiang, B. Zhao, and Q. Peng. Sketch recognition with deep visual-sequential fusion model. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 448–456, 2017.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916, 2015.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. page 1097–1105, 2012.

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[16] K. Li, K. Pang, J. Song, Y.-Z. Song, T. Xiang, T. M. Hospedales, and H. Zhang. Universal sketch perceptual grouping. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 582–597, September 2018.

[17] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[18] L. Liu, G. Pettinati, and C. Y. Suen. Computer-aided Wartegg drawing completion test. In *Proceedings of the International Conference on Pattern Recognition and Artificial Intelligence*, pages 575–580. Springer, 2020.

[19] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[20] N. T. Ly, L. Liu, C. Y. Suen, and M. Nakagawa. Hand-drawn object detection for scoring wartegg zeichen test. In *Proceedings of the International Conference on Pattern Recognition and Artificial Intelligence*, pages 109–114. Springer, 2020.

[21] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[22] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[23] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018.

[24] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 39(06):1137–1149, 2017.

[25] P. Sangkloy, N. Burnell, C. Ham, and J. Hays. The sketchy database: learning to retrieve badly drawn bunnies. *ACM Transactions on Graphics (TOG)*, 35(4):1–12, 2016.

[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[27] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 4278–4284. AAAI Press, 2017.

[28] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[29] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[30] F. Wang, S. Lin, H. Li, H. Wu, T. Cai, X. Luo, and R. Wang. Multi-column point-cnn for sketch segmentation. *Neurocomputing*, 392:50–59, 2020.

[31] E. Wartegg. Gestaltung und charakter. ausdrucksdeutung zeichnerischer gestaltung und entwurf einer charakterologischen typologie. 1939.

[32] P. Xu, T. M. Hospedales, Q. Yin, Y.-Z. Song, T. Xiang, and L. Wang. Deep learning for free-hand sketch: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2022.

[33] P. Xu, Y. Huang, T. Yuan, K. Pang, Y.-Z. Song, T. Xiang, T. M. Hospedales, Z. Ma, and J. Guo. Sketchmate: Deep hashing for million-scale human sketch retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8090–8098, 2018.

[34] Q. Yu, Y. Yang, F. Liu, Y.-Z. Song, T. Xiang, and T. M. Hospedales. Sketch-a-net: A deep neural network that beats humans. *International journal of computer vision*, 122(3):411–425, 2017.

[35] P. Zhang, Y. Zhong, and X. Li. Slimyolov3: Narrower, faster and better for real-time uav applications. pages 37–45, 2019.