# Recognition of Graphological Wartegg Hand-Drawings

Yunqi Xu([✉]) and Ching Y. Suen

Centre for Pattern Recognition and Machine Intelligence (CENPARMI),
Gina Cody School of Engineering and Computer Science, Concordia University,
Montreal, QC H3G 1M8, Canada
{x_yunq,suen}@encs.concordia.ca

**Abstract.** Wartegg Test is a drawing completion task designed to reflect the personal characteristics of the testers. A complete Wartegg Test has eight 4 cm × 4 cm boxes with a printed hint in each of them. The tester will be required to use 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 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 paper, we present a complete framework including PCC (Pearson's Correlation Coefficient) to extract lines and curves, SLIC for the selection of feature key points, DBSCAN for object cluster, and finally YoloV3-SPP model 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.

**Keywords:** Wartegg test · Image processing · Object detection

## 1 Introduction

The Wartegg Test, also called Wartegg Zeichen Test, is a classic psychology test that can reflect personalities of a tester. A Wartegg Zeichen Test form is an A4 paper consisted of eight 4 cm × 4 cm squares in two rows with a simple printed sign in each square [15]. The tester will be required to draw anything in their mind in each square using a pencil. Recognizing testers' drawings inside those boxes correctly can help graphologists to detect the tester's thoughts and predict the tester's potential psychological problem [9]. In our experiment, we followed the above description to collect our Wartegg Zeichen Test forms and then scan

Supported by CENPARMI & NSERC.

them into digital format, finally split one Wartegg Zeichen Test form into eight images.

In recent years, the huge amount of deep learning algorithms for object recognition make it possible to detect those hand-drawn pictures in Wartegg Test images using a computer. However, many problems become obstacles and make the detection of Wartegg Test image set very difficult. The challenges of recognition Wartegg Test images are the same as the other Hand-drawn images recognition tasks [16]. Firstly, these hand-drawn images are abstract, a complex object in real world can be present as a simple shape in hand-drawn images. Second, their colour context is lacking compared with the real images that was taken by camera. Third, they are diverse because different people have different draw style for the same object. However, the challenges of recognition of Wartegg Test are not just limited on these three aspects, Wartegg Test images are also unpredictable, not only the number of objects in each square, but also the size of an object. Usually, for training a detector, we need to input a huge amount of images to make sure the network is robust enought to achieve a high performance when testing it. But current open source hand-drawn image sets, such as QuickDraw [5], Sketchy [13] and Tu-Berlin [3] are all one object images with the same size and located in the center of the images. So, directly using these open source images as training images and testing on Wartegg Test images will not get a satisfactory result.

To make it possible to recognize Wartegg Test images using network, in this paper, we present a complete Wartegg Test image sets process, combine the PCC for the extraction of lines and curves, SLIC + DBSCAN for objects split, these methods solve the problem of different distribution of training image set and Wartegg Test image set and partly increase the robustness of the neural network. Finally, we used a transfer learning based approach to load the pre-trained ImageNet parameters into our YoloV3-SPP backbone network DarkNet53 to increase the converge speed of our network, and then train a YoloV3-SPP for shape detection.

The paper is structured as follows. Section 2 briefly introduces the previous work. Section 3 outlines the concepts of our methods, PCC, SLIC algorithm, DBSCAN algorithm and YoloV3-SPP. Section 4 describes the experimented details, and Sect. 5 for the conclusion and future work of our experiment.

## 2 Background

The begining of applying classification and recognition deep learning model to hand-drawn images can be retrospect from 2012. In this year, Tu-Berlin [3] presented their hand-drawn image set. After this, in 2016 Sketchy images have been presented [13] with 125 categories. In 2017, Google presented one of the largest sketch image set QuickDraw [5]. And in 2018, image sets SPG [8] and SketchSeg [14] were publised. Finally, in 2019, our Warteg Test image set [9] has been collected which contains 30 categories and more than 900 images. Accompanied with this image sets, some famous deep learning models also have been mentioned in the past ten years, such as Sketch-net [18], Deep visual sequential fusion net [6], SketchRNN [5] and Sketchmate [17]. Although these models achieve high performance, they are tested on Tu-Berlin and QuickDraw, some single object image sets. However, free hand-drawn images should be diverse, like our Wartegg Test image set which contains many images that have multiple objects in one image. So it is important to explore how to detect multiple objects using current open source image set.

## 3 Method

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

### 3.1 Image Processing

Image processing includes two parts. Firstly, using PCC (Pearson's Correlation Coefficient) [2] to extract lines and curves and remove slated noises caused by image format transformation. Secondly, utilizing the SLIC superpixel and DBSCAN cluster algorithm to extract every object in the image and delete meaningless parts.

**PCC for Feature Selection.** Donati [2] used PCC to extract the features of sketch images based on the relevance of the pixels in this image. Using this method, those pixels neither belong to lines nor curves can be deleted.

Firstly, define

$$kernel = Gaussian\_kernel(next\_odd(7 * \sigma_i, \sigma_i)) \tag{1}$$

where, $\sigma_i = C * \sigma_{i-1} = C^i * (w_{min}/b), i \in [1, \log_c(w_{max}/w_{min}) - 1]$, and $w_{min}$ is the minimum line width, $w_{max}$ is the maximum, C and b are constant, usually we will use 2 and 3 separately.

After generating the $i^{th}$ Gaussian kernels Using Eq. 1 we will use

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

to calculate several PCC images, finally, we will use

$$MPCC = \begin{cases} maxPCC_{xy}, & \mid maxPCC_{xy} \mid > \mid minPCC_{xy} \mid \\ minPCC_{xy}, & otherwise \end{cases} \tag{3}$$

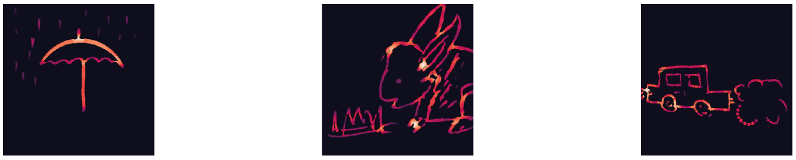
**SLIC Algorithm for Superpixel Split.** SLIC was mentioned by R.Achanta [1], which splits one image into many small sub patches (also called superpixels) based on their values and locations.

The algorithm of SLIC is presented in Algorithm 1:

**Algorithm 1.** SLIC algorithm

```
Input: $k$: number of superpixels ; $m$; $iterations$; $image\_width$; $image\_height$
Initialization:
    $S = \sqrt{(image\_width * image\_height)/k}$; $centerset = C_k = (l_k, x_k, y_k)_{k=1...k}$;
    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 $p_i.cluster = None$, $p + i.distance = +\infty$, $i = 1$
1: if i $\leq$ iteration then
2:     for each cluster $c_k$ in clusterset do
3:         for each point $p_i$ in $[c_k.x - s : c_k.x + s \ ; c_k.y - s : c_k.y + s]$ do
4:             $D = \sqrt{(\sqrt{(p_i.l - c_k.l)^2}/m)^2 + (\sqrt{(p_i.x - c_k.x)^2 + (p_i.y - c_k.y)^2}/s)^2}$
5:             if D < $p_i.distance$ then
6:                 $p_i.distance$ = D
7:                 $p_i.cluster$ = $c_k$
8:     $i = i + 1$;
```

**DBSCAN for Object Segmentation.** DBSCAN was mentioned by M.Ester [4], which has been designed to discover clusters with arbitrary shape based on the density of dataset. The main idea of 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 density. This idea is similar to our Wartegg Test image set, the density of lines and curves is usually higher in an object, e.g. Fig. 1.



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

**Fig. 1.** Examples of Density distribution of Wartegg Test images

We will calculate the ratio of the number of pixels that have a value in a 5 * 5 window, and generate these density distribution images. The color in Fig. 1 which is close to white means the density score is high, on the other hand, when the color is black, it means the density score is approximately zero. These images agree with our conjecture, so we can utilize DBCANS to cluster objects in the same image.

After using Superpixel to split an image into numerous small sub-patches, for each patch, we keep those pixels at the center of the sub-patch located on the lines or curves. Based on previous tests, the remaining key points belong to the same object should be much closer, and for those points from different objects, they should have a different distribution. Also, points belonging to the noisy part should have a very low density.

The algorithm of DBSCAN is shown below:

**Algorithm 2.** DBSCAN algorithm

```
Input: $Eps$, $Minpts$, $Dataset$
Initialization: $clusterid = 0$ ; $\forall$ point $p_i$, $p_i \in$ dataset, $p_i$.cluster = None
 1: for each point $p_i$ in dataset do
 2:   if $p_i$.cluster is None then
 3:     clusterset = [ $\exists\, p_j$, if dist($p_i$, $p_j$) $\leq$ Eps]
 4:     if clusterset.length < Minpts then
 5:       $p_i$.cluster is Noise
 6:     else
 7:       for each point $p_c$ in clusterset do
 8:         $p_c$.cluster = clusterid ; Remove $p_i$ from clusterset
 9:       while cluster is not Empty do
10:         $p_{new}$ = clusterset[0];$new\_cluster$ = [ $\exists\, P'$, if dist($p_{new}$, $p'$) $\leq$ Eps]
11:         if $new\_clusterset$.length $\geq$ Minpts then
12:           for each point $p'_{new}$ in $new\_clisterset$ do
13:             if $p'_{new}$.cluster is None then
14:               clusterset.append($p'_{new}$); $p'_{new}$.cluster = clusterid
15:             if $p'_{new}$.cluster is Noise then
16:               $p'_{new}$.cluster = clusterid
17:   clusterid += 1
```

### 3.2 YoloV3-SPP

The model we choose for the object detection task is YoloV3-SPP [19] which is the third version of the Yolo model with an SPP structure [7] that can improve the precision of the original YoloV3. Figure 2 presents the overall structure of YoloV3-SPP. It contains a backbone DarkNet53 [12] which is composed of many Residual blocks and a SPP structure, finally the output will be generated through three different feature maps for predicting different scales of an object. Compared with the first output directly follows the SPP structure and the Convolutional set (the first branch in Fig. 2), the second output and the last output comes from the concatenated (the green ball) result of a Residual $\times$ 8 block and a Upsample layer (the red box) accompany with a Convolutional set block.

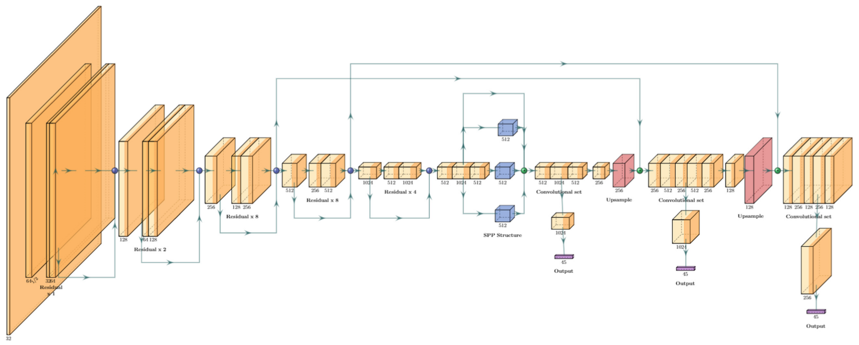


**Fig. 2.** The structure of YoloV3-SPP (Color figure online)

## 4 Experiment

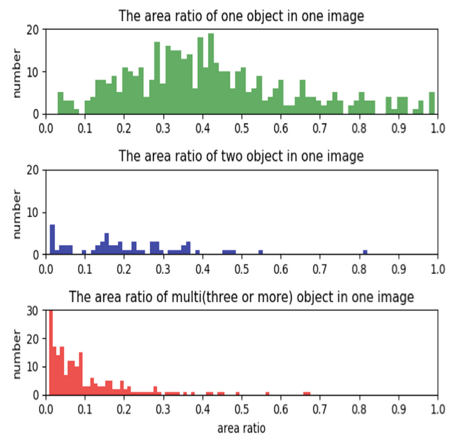
Our experiment will follow the previous methods introduced in Sect. 3.1 to process our Wartegg Test image set. We will firstly introduce our training and test image sets in Sect. 4.1, and then discuss the pixel value augment in Sect. 4.2, describe the details of image processing result in Sect. 4.3, and finally analyze the object detection result in Sect. 4.4.

### 4.1 Introduction of Dataset

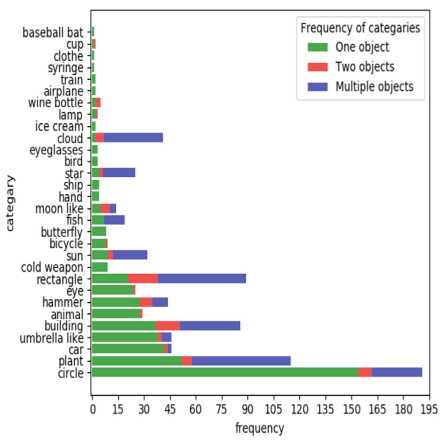
During our experiment, we used QuickDraw [5] as the training image set and our collected Wartegg Test image set as the testing input to validate the performance of the YoloV3-SPP model.

**Testing Image Set.** Our Wartegg Zeichen Test form are collected by Liu [9] in 2019. Firstly, we labeled all objects in each image, and then, merged together those categories that share the same pattern. For example, face, circle, cookie and tire, apple are circular patterns, they can be treated as the shape circle. Also, Parachute, hot air balloon and Umbrella all have a curved ceiling at the top and converging tail at the bottom.

Figure 3a 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, and multiple objects, there are more tiny objects in each image.



(a) Area Ratio



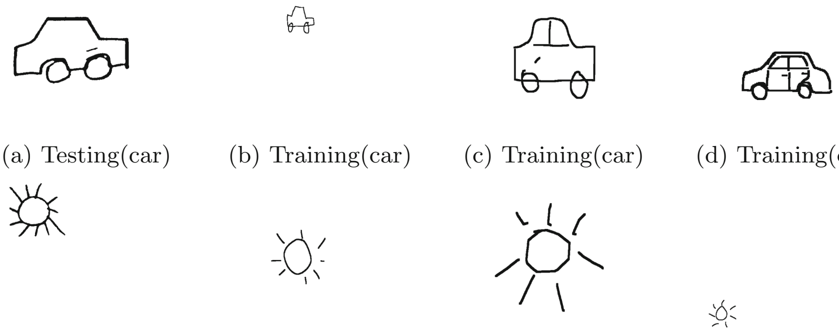
(b) Frequency of Categaries

**Fig. 3.** Examples of Density distribution of Wartegg Test images

After labeling and merging, we counted the frequency of each category as shown in Fig. 3b. Finally, we will select the top 11 classes frequently appeared in those images that contain only one object. The 11 classes are: circle, planet, car, umbrella, building, animal hammer, eye, rectangle, cold weapon and sun. Also, We have 439 images with a single object, 55 images have two objects and 93 images contain multiple objects.

**QuickDraw Image Set.** The QuickDraw image set [5] has been collected through the game "QuickDraw" developed by Google, which has very similar rules as "Drawize", instead drawing something and guessed by friend, Quick-Draw lets the tester draw an object based on a word hint, and the machine will guess what the tester has drawn. The QuickDraw image set contains more than 300 categories which is suitable for us to select the useful categories.

We generate the training image set based on the area distribution of 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. Since QuickDraw was drawn by computer mouse or touch pad, electronic pencil, the width of its lines and curves is only one pixel, but the lines and curves of Wartegg Test image set are drawn by pencil, so their width varies from 4 to 8 pixels. Then, we will enlarge the width of lines and curves in QuickDraw to 6 pixels. Figure 4 shows how we generate training images with different scales and bold lines and curves of the training image set that make them suitable for the testing image set.



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

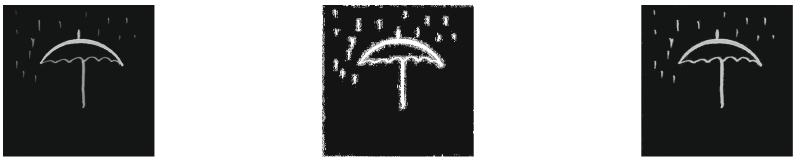
(e) Testing(Sun) (f) Training(Sun) (g) Training(Sun) (h) Training(Sun)

**Fig. 4.** 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. 5 shows, although the original Fig. 5a looks good, there are lots of noises in the image surrounding the lines and curves.



(a) original image (b) Actual (c) after augment

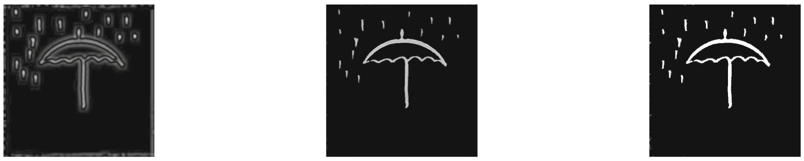
**Fig. 5.** Actual image with noise

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 will normalize the value of pixels between (0, 1), and then follow the steps described in Sect. 3.1 to extract lines and curves.

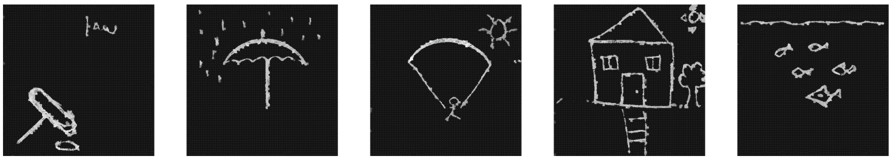
Figure 6a presents each filter step in the PCC process during our experiment. Figure 6b shows the result after PCC algorithm. Finally, Fig. 6c shows the denoised result. Compared with Fig. 5, those noises surrounding the lines and curves with unpredictable values are deleted.



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

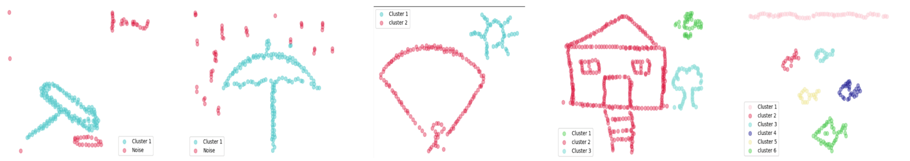
**Fig. 6.** PCC result

However, not only the salt and pepper noise pixels are noises, during our detection process, those pixels with useless information also need to be treated as noise since this will impact our final inference result. So, we will split one whole image into many small superpixels (Fig. 7). Then, we kept those center points of a small pitches that located on strokes as our key points and use DBSCAN mentioned in Sect. 3.1 to classify these key points into multiple clusters.



(a) one object (b) one object (c) two objects (d) three objects (e) multi objects

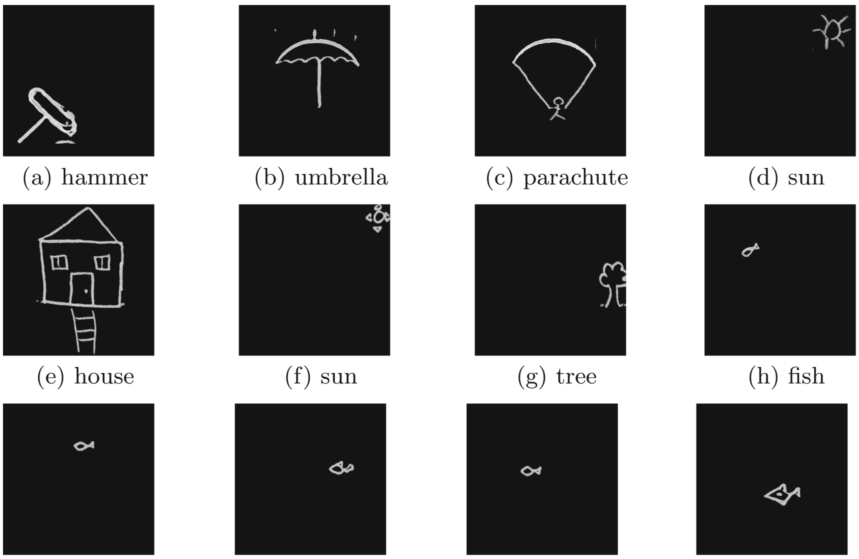
**Fig. 7.** SLIC result



(a) one object (b) one object (c) two objects (d) three objects (e) multi objects

**Fig. 8.** DBSCAN cluster result

Figure 8 shows the cluster result. Finally, based on the cluster result, we split each image into multiple images with only one object. Figure 9 shows the result. So, for an image which contains N objects, we will get N new images and one object per image in the end. For example, in Fig. 8d, we have three objects in the image: a house, a tree and a sun, after splitting, we will get Fig. 9e, Fig. 9f and Fig. 9g. By doing so, the Wartegg Test image set will have the same distribution as the training image set.



(a) hammer (b) umbrella (c) parachute (d) sun

(e) house (f) sun (g) tree (h) fish

(i) fish (j) fish (k) fish (l) fish

**Fig. 9.** Split result

## 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 the transfer learning based approach that firstly loads the parameters of pre-trained DarkNet53 on ImageNet and fine-tune the last layer using the original QuickDraw images to make the backbone network learn the features of sketch images. The transfer learning can help us solve the problem of insufficient number of training images and increase the converging speed. In our experiment, loading the pre-trained parameters from well-trained DarkNet53 on the ImageNet, then fine-tune the last layer will spend less time, usually 5 epoches to converage, compared with directly training a DarkNet53 on QuickDraw images, which need around 15 epoches to let the network converage, and around 20 epoches to let the result of the network to be stable. Once the training process of DarkNet53 is completed, we load these parameters without the last fully Connected Layer of DarkNet53 into the YoloV3-SPP Network. Using our generated QuickDraw image set we train the YoloV3-SPP for the detection task.

During the testing process, We have three image sets, Image set 1 contains those images that the original images have only one object, Image set 2 contains those images which the original images have one or two object, and Image set 3 contains those images that the original images have one, two or multiple objects. We input each split Wartegg Test image into the trained YoloV3-SPP model to get the inference result. We will get the inference results for each image, then, we will merge those split images which belong to individual images together with their predicted bounding boxes. So, 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.** 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% |

Table 1 shows that we got a much better result compared with previous result, which got 68.72% (IoU = 0.5) for one object detection [10], and we increase this score to 87.9% (IoU = 0.5) with an mAP of 67.7%. However, the result drops when testing on one and two 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 because with an increase in the object number, the area ratio of each object with its image size will decrease. On the other hand, for multiple object detection, there are more tiny objects in the testing image set compared with one object image set. As Fig. 3a shows, these tiny objects will lower the performance of the network. On the other hand, as a comparision, we also use the original images as the testing image set to verify the performance of the trained detector on original Wartegg Test image set. The result has decreased significantly, especially for image sets 2 and 3. These may be because in the training image set, each image has one object, but in our Wartegg test image set, an image contains an indeterminate number of objects, also if an image contains more than one object, the distance between two objects is short, the target area inside a bounding box may contain two objects, and one object could be treated as the interference part when another object is detected.

## 5 Conclusion and Future Work

In this paper, we present a complete framework of the Wartegg Test image set in object detection. Our method becomes a bridge between one 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 still are some limitations of 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 very suitable for tiny object detection [11], the next step may need to design part of the network which is suitable for both large and tiny objects to improve the performance.

## References

1. Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süsstrunk, S.: SLIC superpixels compared to state-of-the-art superpixel methods. IEEE Trans. Pattern Anal. Mach. Intell. **34**(11), 2274–2282 (2012). https://doi.org/10.1109/TPAMI.2012.120
2. Donati, L., Cesano, S., Prati, A.: A complete hand-drawn sketch vectorization framework. Multimed. Tools Appl. **78**(14), 19083–19113 (2019). https://doi.org/10.1007/s11042-019-7311-3
3. Eitz, M., Hildebrand, K., Boubekeur, T., Alexa, M.: Sketch-based image retrieval: benchmark and bag-of-features descriptors. IEEE Trans. Vis. Comput. Graph. **17**(11), 1624–1636 (2011)
4. Ester, M., Kriegel, H.P., Sander, J., Xu, X., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD, vol. 96, pp. 226–231 (1996)
5. Ha, D., Eck, D.: A neural representation of sketch drawings. In: International Conference on Learning Representations (2018)
6. He, J.Y., Wu, X., Jiang, Y.G., Zhao, B., Peng, Q.: Sketch recognition with deep visual-sequential fusion model. In: Proceedings of the 25th ACM International Conference on Multimedia, pp. 448–456 (2017)
7. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Trans. Pattern Anal. Mach. Intell. **37**(9), 1904–1916 (2015). https://doi.org/10.1109/TPAMI.2015.2389824
8. Li, K., et al.: Universal sketch perceptual grouping. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11212, pp. 593–609. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01237-3_36
9. Liu, L., Pettinati, G., Suen, C.Y.: Computer-aided Wartegg drawing completion test. In: Lu, Y., Vincent, N., Yuen, P.C., Zheng, W.-S., Cheriet, F., Suen, C.Y. (eds.) ICPRAI 2020. LNCS, vol. 12068, pp. 575–580. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-59830-3_50
10. Ly, N.T., Liu, L., Suen, C.Y., Nakagawa, M.: Hand-drawn object detection for scoring Wartegg Zeichen test. In: Lu, Y., Vincent, N., Yuen, P.C., Zheng, W.-S., Cheriet, F., Suen, C.Y. (eds.) ICPRAI 2020. LNCS, vol. 12068, pp. 109–114. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-59830-3_9
11. Nguyen, N.-D., Do, T., Ngo, T.D., Le, D.-D.: An evaluation of deep learning methods for small object detection. J. Electr. Comput. Eng. **2020**, 1–18 (2020). https://doi.org/10.1155/2020/3189691. Article ID 3189691
12. Redmon, J., Farhadi, A.: YOLOv3: an incremental improvement. arXiv (2018)
13. Sangkloy, P., Burnell, N., Ham, C., Hays, J.: The sketchy database: learning to retrieve badly drawn bunnies. ACM Trans. Graph. (TOG) **35**(4), 1–12 (2016)
14. Wang, F., et al.: Multi-column point-CNN for sketch segmentation. Neurocomputing **392**, 50–59 (2020)

15. Wartegg, E.: Gestaltung und charakter. ausdrucksdeutung zeichnerischer gestaltung und entwurf einer charakterologischen typologie (1939)
16. Xu, P., Hospedales, T.M., Yin, Q., Song, Y.-Z., Xiang, T., Wang, L.: Deep learning for free-hand sketch: a survey. IEEE Trans. Pattern Anal. Mach. Intell. 1. https://doi.org/10.1109/TPAMI.2022.3148853.
17. Xu, P., et al.: SketchMate: deep hashing for million-scale human sketch retrieval. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8090–8098 (2018)
18. Yu, Q., Yang, Y., Liu, F., Song, Y.Z., Xiang, T., Hospedales, T.M.: Sketch-a-net: a deep neural network that beats humans. Int. J. Comput. Vis. **122**(3), 411–425 (2017). https://doi.org/10.1007/s11263-016-0932-3
19. Zhang, P., Zhong, Y., Li, X.: SlimYOLOv3: narrower, faster and better for real-time UAV applications. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops, October 2019