A Comparison of Object Detection Methods on Circuit Sketches

1st Aras Edeş
Faculty of Electrical and Electronics Engineering
Istanbul Technical University
Istanbul, Turkey
edes20@itu.edu.tr

***Abstract*—Circuit sketches are widely used in various applications. However, converting these sketches into a digital format for further analysis or storage is a laborious and time-consuming task. Automating this process requires the recognition of components within these diagrams. This paper presents an evaluation of different deep learning-based object detection models using a publicly available comprehensive circuit sketch dataset. The primary objectives are to assess the performance of promising models from previous studies and establish a baseline for future research. Additionally, this paper introduces the utilization of DETR (Detection Transformers) for circuit component detection.**

***Index Terms*—engineering diagram, circuit schematic, object detection, DETR, YOLO, Faster RCNN, CGHD**

I. Introduction

Circuit sketches play a crucial role in numerous circuit design and analysis workflows. These sketches are utilized by students in educational settings, as well as professionals and hobbyists, for prototyping and converting schematics into a digital format. Furthermore, circuit sketches are essential for computer-aided analysis of circuits, highlighting the need for their digitization.

Previous methods employed traditional statistical models and various heuristics such as HOG transform, SVM, and KNN [1, 2]. However, with the advancements in deep learning, the adoption of deep learning models for circuit sketch recognition, particularly for object detection, has gained significant momentum. Demonstrations have been made using models such as YOLOv3, YOLOv5, SSD300, SSD512, Faster R-CNN, and custom models [3, 4]. However, a major challenge with these demonstrations lies in the inadequacy of available datasets.

Circuit sketch recognition can be categorized into two stages: component detection and connective element detection. This study focuses on exploring alternative approaches for component detection and establishing a baseline for future research.

To facilitate a comprehensive comparison, various CNN-based single-stage, two-stage, and multi-stage models will be employed. Additionally, this study introduces Transformer-based models for the detection of circuit components, which, to the best of our knowledge, has not been previously explored. The performance evaluation will include RetinaNet, Faster RCNN, Cascade Mask RCNN, YOLOv5, and DETR [5, 6, 7, 8, 9].

Code available at "https://github.com/Ar-Ed/CGHD_Detection"

II. Dataset

The primary objective of this study is to establish a baseline and evaluate various methods for object detection using a realistic dataset of circuit sketches. Previous studies' datasets lack comprehensiveness in terms of circuit types, mainly consisting of overly simplistic circuits, and they are not publicly accessible. Consequently, this study utilizes the Circuit Graph Hand Drawn (CGHD) [10] dataset for the aforementioned reasons.

A. CGHD

CGHD comprises 45 different types of objects, encompassing a wide range of passive and active circuit elements, digital gates, integrated circuits, and interface devices. Additionally, annotations are provided for junctions, crossovers, terminals, and text elements. CGHD consists of 1,152 images, containing a total of 48,563 annotations, out of which 33,856 correspond to junctions, texts, and crossovers.

In [10], it is recommended to use 125 images for training, 7 for validation, and 12 for testing. A baseline mean Average Precision (mAP) of 52% is reported using a Faster R-CNN model with a ResNet152 backbone. For consistency, the same split sizes will be employed in this study. However, it is important to note that this split methodology may result in a mismatch between the train, test, and validation sets. Different halves of the 45 classes are not equally represented in both the test and train sets.

III. Models

In our experiments, we selected YOLOv5 as the single-stage detector, RetinaNet and Faster R-CNN as two-stage detectors, and Cascade R-CNN as the multi-stage detector. DETR can also be considered as a single-stage detector. The CNN models employ either ResNet50 or ResNet101, or both, as the backbones.

RetinaNet, Faster R-CNN, and Cascade R-CNN are implemented using the Detectron2 framework. DETR is built

using the DETR library, which utilizes the Detectron2 API. We utilize the small YOLOv5 implementation from Ultralytics as the YOLO-based detector. The Faster R-CNN and RetinaNet implementations in Detectron2 additionally utilize Feature Pyramid Networks (FPN) during the feature extraction process to generate semantically strong features while preserving resolution.

A. Faster R-CNN

Faster R-CNN replaces the selective search algorithm with Region Proposal Networks (RPN). By sharing convolutional layers with the object detection network, RPN improves both inference and training latency.

B. RetinaNet

RetinaNet introduces the focal loss function to address the foreground-background class imbalance issue in single-stage detectors. It incorporates feature pyramids, anchor boxes, and focal loss to emphasize harder examples during training.

C. Cascade R-CNN

Cascade R-CNN trains and tests on multiple IoU threshold levels to reduce overfitting and improve the detection of robust objects. In this study, Cascade Faster R-CNN is employed, although other base models can also be cascaded.

D. YOLOv5

YOLOv5 is an alternative detector that differs from sliding window or region proposal-based detectors. YOLOv5 does not repurpose classifiers for building detectors, which was a popular approach in the past. While most detectors struggle with detecting small objects, this is particularly challenging for YOLOv5.

E. DETR

Unlike CNN detectors that utilize the NMS algorithm and proposals or anchors, DETR does not rely on handcrafted methods. One of the distinguishing features of DETR is its ability to leverage global relationships between objects and the background. It achieves this by employing transformers on the features extracted by the CNN backbone.

IV. Methodology

The aforementioned models are trained on the CGHD dataset using the split previously described. Faster R-CNN and RetinaNet are trained with both ResNet101 and ResNet50 backbones. Faster R-CNN, Cascade R-CNN, and RetinaNet models are pre-trained on the COCO 2017 dataset for approximately 37 epochs. Similarly, DETR and YOLOv5 models are pre-trained on the COCO 2017 dataset.

The evaluation metrics used are mean average precision (mAP) on the test, validation, and train sets. Although the models may exhibit bias towards the training set (including the validation set due to early stopping), evaluating their performance on the train set provides a measure of overall performance across all classes.

A. Training Details

The libraries used in this study come with default image augmentations, which have been disabled to minimize the impact of library-specific performance differences. Default hyperparameters provided by the libraries are generally preferred, with the exception of learning rates, epochs, and batch sizes. All images are resized to 1024 pixels along the longer dimension while maintaining the aspect ratio. Smaller image sizes resulted in objects with insufficient pixel counts (less than 5 pixels).

B. Evaluation

In object detection, the model's objective is to localize objects using bounding box coordinates and classify the objects. The mean average precision (mAP) score is the standard metric for evaluating these two objectives together. The mAP score is built upon precision, recall, and intersection over union (IoU) measures.

Precision and recall are common classification metrics defined as follows:

$$P = \frac{TP}{TP + FP} \tag{1}$$

$$R = \frac{TP}{TP + FN} \tag{2}$$

Intersection over union measures the relative overlap of predicted bounding boxes. In the context of object detection, given two bounding boxes (B_1 as the ground truth box and B_2 as the predicted box), IoU is defined as follows:

$$IoU = \frac{\mid B_1 \cap B_2 \mid}{\mid B_1 \mid + \mid B_2 \mid - \mid B_1 \cap B_2 \mid} \tag{3}$$

Average precision (AP) is the area under the precision-recall curve for varying classification probabilities. Mean average precision (mAP) is the counterpart of average precision for multi-class classification. In the context of object detection, the IoU is another parameter for mean average precision, and mAP scores are generally reported with corresponding IoU scores (e.g., mAP@0.5). For this study, mean average precision with an IoU of 0.5 and the range of 0.5 to 0.95 will be referred to as AP@0.5 and AP@0.5:0.95, respectively.

V. Results

In this study, we conducted training and evaluation using a single NVIDIA RTX 3060 6144MB GPU. The validation mean Average Precision (mAP) scores of the Cascade Faster R-CNN model are plotted in Figure 1 and Figure 2. The mAP scores for the Faster R-CNN and RetinaNet models with different backbones are plotted in Figure 3, Figure 4, Figure 5, and Figure 6, respectively. The DETR model's mAP scores are shown in Figure 7 and Figure 8, while YOLOv5 small is presented in Figure 9 and Figure 10. Lastly, the YOLOv8 large model is displayed in Figure 11 and Figure 12. In all the figures, the best scores achieved by each model are indicated. The models were selected based on early stopping with respect to their mAP@0.5 score, except for YOLOv8.

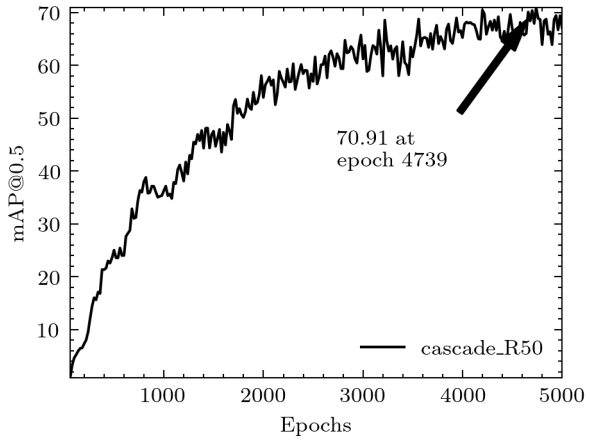

Fig. 1. Evolution of mAP@0.5 during training for Cascade Faster RCNN with ResNet50 Backbone and FPN

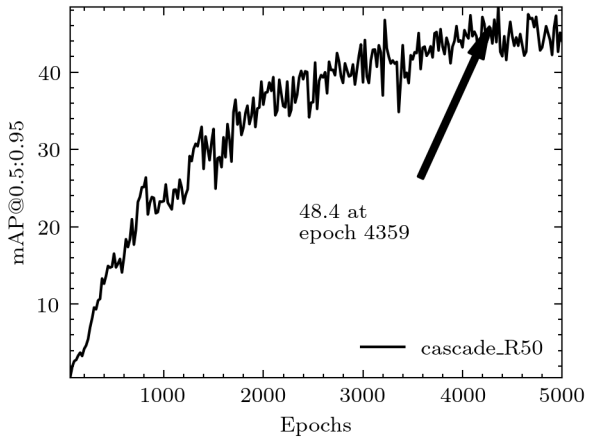

Fig. 2. Evolution of mAP@0.5:0.95 during training for Cascade Faster RCNN with ResNet50 Backbone and FPN

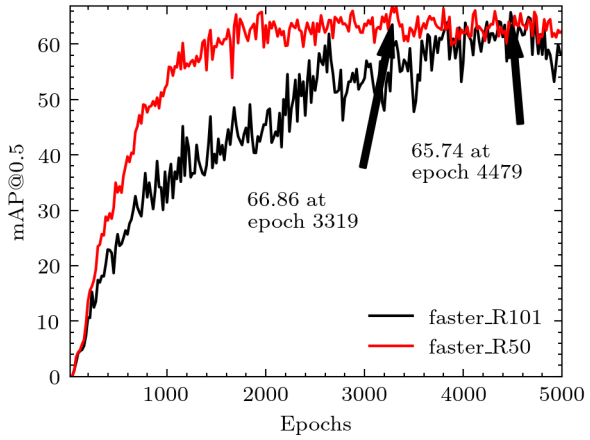

Fig. 3. Evolution of mAP@0.5 during training for Faster RCNN with ResNet50 and ResNet101 Backbones and FPN

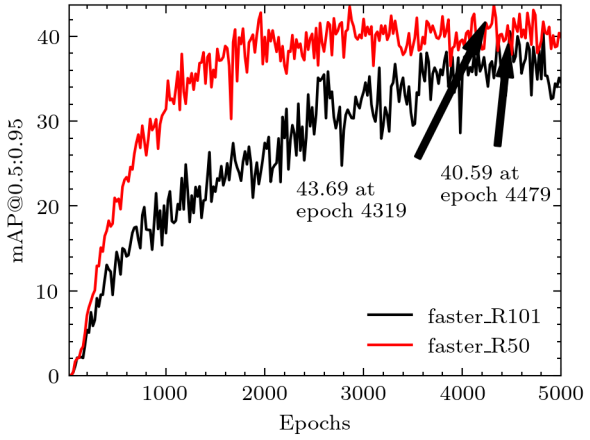

Fig. 4. Evolution of mAP@0.5:0.95 during training for Faster RCNN with ResNet50 and ResNet101 Backbones and FPN

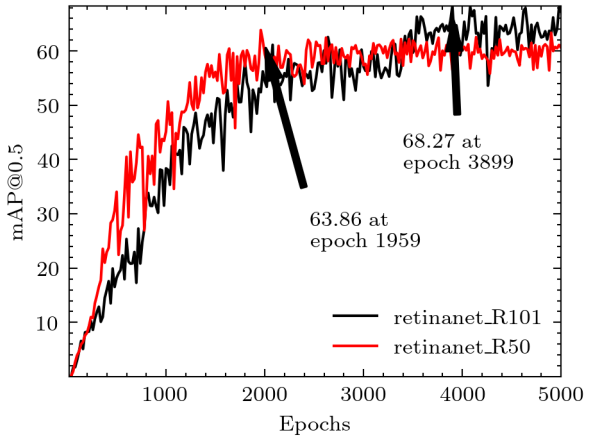

Fig. 5. Evolution of mAP@0.5 during training for RetinaNet with ResNet50 and ResNet101 Backbones and FPN

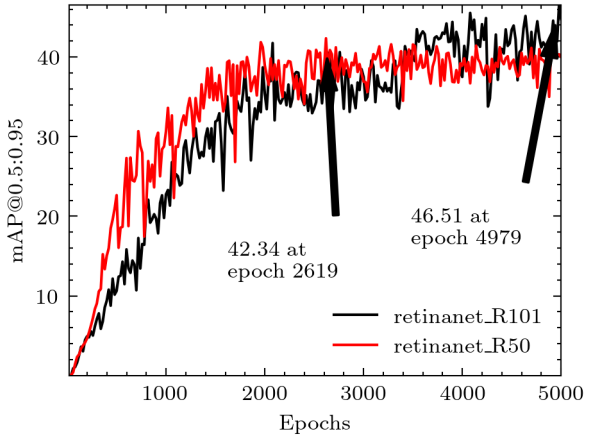

Fig. 6. Evolution of mAP@0.5:0.95 during training for RetinaNet with ResNet50 and ResNet101 Backbones and FPN

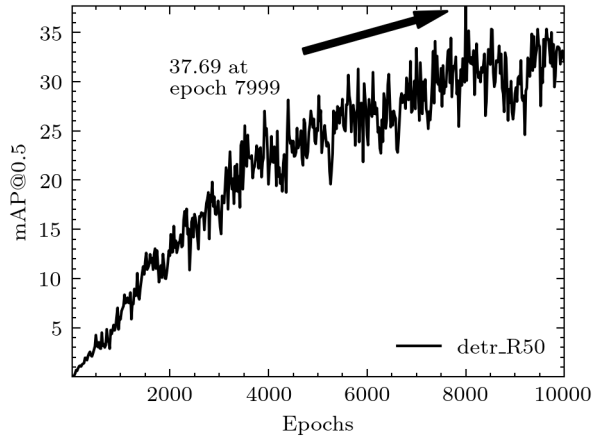

Fig. 7. Evolution of mAP@0.5 during training for DETR with ResNet50 Backbone

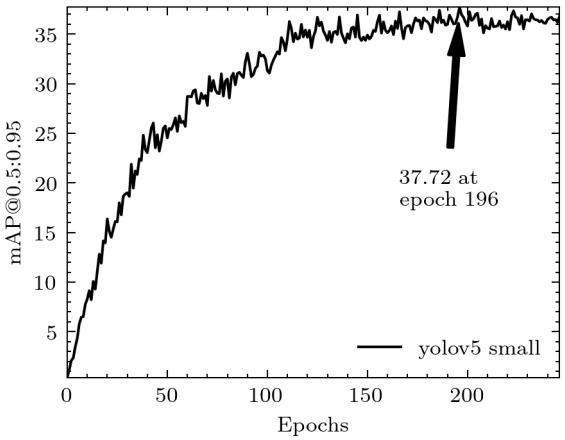

Fig. 10. Evolution of mAP@0.5:0.95 during training for YOLOv3 small

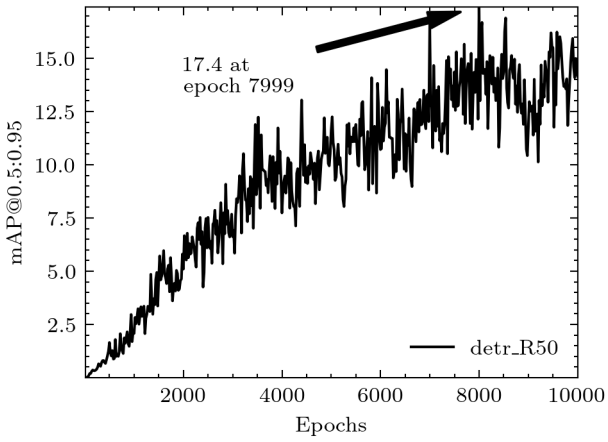

Fig. 8. Evolution of mAP@0.5:0.95 during training for DETR with ResNet50 Backbone

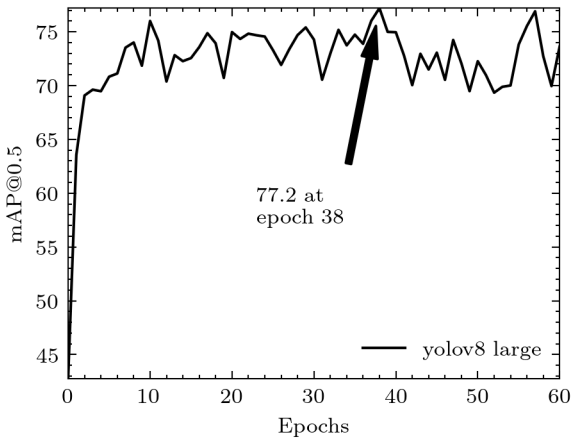

Fig. 11. Evolution of mAP@0.5 during training for YOLOv8 large

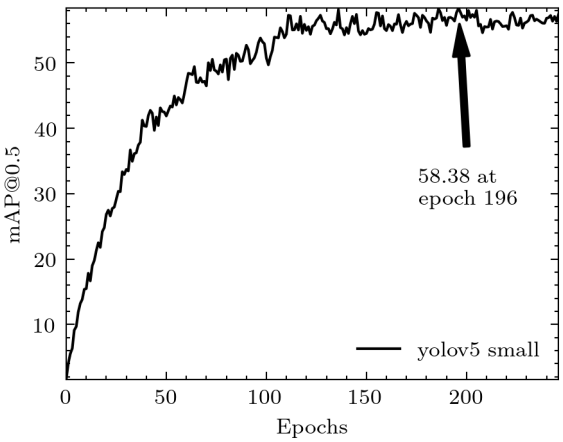

Fig. 9. Evolution of mAP@0.5 during training for YOLOv3 small

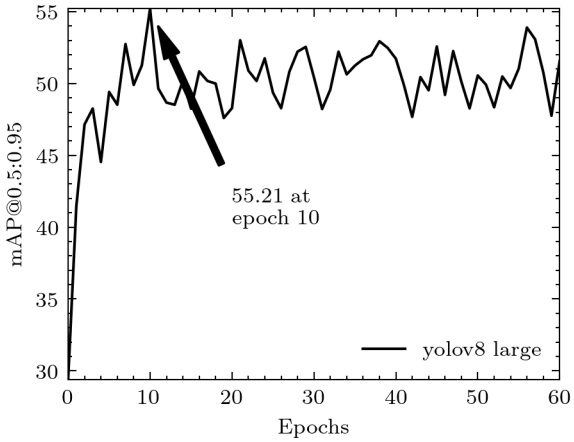

Fig. 12. Evolution of mAP@0.5:0.95 during training for YOLOv8 large

On table I, table II, table III mAP@0.5:0.95 and mAP@0.5 scores of final models are provided for the train set, test set and validation set.

TABLE I
TRAIN

	map@0.5:0.95	***map@0.5***
faster rcnn r50	69.29	94.04
faster rcnn r101	43.04	64.8
cascade rcnn r50	59.38	74.15
retinanet r50	75.77	93.81
retinanet r101	64.5	80.3
yolov5s	-	-
yolov8l	80.9	97.6
detr	48.8	79.86

TABLE II
TEST

	map@0.5:0.95	***map@0.5***
faster rcnn r50	49.53	74.37
faster rcnn r101	44.47	73.4
cascade rcnn r50	56.6	80.39
retinanet r50	53.58	74.18
retinanet r101	48.8	70.44
yolov5s	-	-
yolov8l	52.64	69.41
detr	26.84	52.17

TABLE III
VALIDATION

	map@0.5:0.95	***map@0.5***
faster rcnn r50	42.16	67.45
faster rcnn r101	40.9	66.42
cascade rcnn r50	47.26	71.16
retinanet r50	41.97	63.36
retinanet r101	44.4	68.17
yolov5s	-	-
yolov8l	55.32	76.06
detr	22.11	46.16

VI. CONCLUSION

This study conducted an evaluation of various deep learning-based object detection methods for the detection of electrical components and connective components in circuit schematics. Our best-performing model achieved a mean average precision (mAP) of 49%, which is consistent with the mAP of 52% reported in the original CGHD paper [10], taking into account data augmentation, batch size, and epochs. Specifically, RetinaNet with a ResNet101 backbone, Faster R-CNN with a ResNet101 backbone, and Cascade Faster R-CNN exhibited continuous improvement throughout the training process. However, the performance of DETR, although showing improvement in mAP, was relatively lower compared to the other models. To further enhance the performance, future work may involve longer training steps, a strategy to find optimal learning rate for each model, adjustment of average RGB pixel values to align with the CGHD dataset, and additional hyperparameter tuning.

ACKNOWLEDGMENT

Thanks to the lecturer of this class "İsa Yıldırım", and one of the authors of the original CGHD paper "Johannes Bayer" for answering my questions regarding the dataset.

REFERENCES

[1] Dey, M., Mia, S.M., Sarkar, N., Bhattacharya, A., Roy, S., Malakar, S., Sarkar, R. A two-stage CNN-based hand-drawn electrical and electronic circuit component recognition system. Neural Computing and Applications. 33, (2021). https://doi.org/10.1007/s00521-021- 05964-1.

[2] Roy, S., Bhattacharya, A., Sarkar, N., Malakar, S., Sarkar, R. Offline hand-drawn circuit component recognition using texture and shape-based features. Multimedia Tools and Applications 79, (2020). https://doi.org/10.1007/s11042-020-09570-6.

[3] Rachala, R.R., Panicker, M.R. Hand-Drawn Electrical Circuit Recognition Using Object Detection and Node Recognition. SN COMPUT. SCI. 3, 244 (2022). https://doi.org/10.1007/s42979-022-01159-0

[4] MAJEED, M. A., Almousa, T., Alsalman, M., and YOSEF, A. (2020). Sketic: a machine learning-based digital circuit recognition platform. Turkish Journal of Electrical Engineering and Computer Sciences, 28(4), 2030-2045.

[5] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, 'End-to-End Object Detection with Transformers', in Computer Vision – ECCV 2020, 2020, pp. 213–229.

[6] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, 1 June 2017, doi: 10.1109/TPAMI.2016.2577031.

[7] T. -Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, "Focal Loss for Dense Object Detection," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 42, no. 2, pp. 318-327, 1 Feb. 2020, doi: 10.1109/TPAMI.2018.2858826.

[8] Jocher, G., Nishimura, K., Mineeva, T., Vilariño, R. YOLOv5 (2020), https://github.com/ultralytics/yolov5, last accessed 2020/10/10.10.1109/TPAMI.2018.2858826.

[9] Z. Cai and N. Vasconcelos, "Cascade R-CNN: Delving Into High Quality Object Detection," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 2018, pp. 6154-6162, doi: 10.1109/CVPR.2018.00644.

[10] Thoma, F., Bayer, J., & Li, Y. (2021). A Public Ground-Truth Dataset for Handwritten Circuit Diagram Images. ArXiv, abs/2107.10373.

[11] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, 'Detectron2', 2019. [Online]. Available: https://github.com/facebookresearch/detectron2.