

# Document Layout Analysis via Machine Learning SCSE01

Tan Yi Kai, Low Wei Sheng

Hwa Chong Institution (College Section)

Dr Loke Yuan Ren

School of Computer Science and Engineering

**Abstract** - With increasing digitalisation, digital documents are increasingly preferred over physical paper-based documents due to easier organization and access of data. Hence, the large amount of physical documents already available needs to be digitalised to enable data to be retrieved from them easily. These documents often contain information stored in tables, and detecting these tables reliably can be difficult. This is especially the case for semi-bordered and borderless tables, due to the wide variety of table formats. In addition, tables may be highly similar to other parts of a document image, making it even harder. We therefore present an improved method based on object detection that uses iterative transfer learning to achieve better results. Our method utilizes the state-of-the-art general object detection algorithm, Faster RCNN, in order to maximize our performance. It also exploits the table's rectangular features by using rectangular convolutions. This customizes Faster RCNN, which was intended for general object detection, for table detection instead. This helps to improve its performance by allowing it to extract relevant table features even at the early stages. Our proposed method was evaluated on a publicly available online dataset of semi-bordered and borderless tables yielding significantly better results with F1-measure of 86.27%.

**Keywords:** Table Detection, Machine Learning, Document Layout, Convolutional Neural Networks, Transfer Learning

## 1 INTRODUCTION

With increasing automation and digitalisation of many processes, the usage of digital documents instead of physical paper-based documents has been increasing rapidly.<sup>[1]</sup> The large amount of physical documents already available needs to be digitalised to enable data to be retrieved from them easily.

Tables often contain the most important information in documents as they serve to display information systematically<sup>[2]</sup>. However, they first need to be detected before further processing to extract its contents.

The problem of table detection can be rather challenging due to the myriad of ways a table can be constructed. In particular, the detection of semi-bordered or borderless tables proves to be even more difficult, as their features may be difficult to distinguish from other charts or other margins in the

document, as seen in Figure 1. In the example of Figure 1, the other margins, such as those between paragraphs of text, may confuse the model.

Year	1970-1979	1980-1989	1990-1999
1970-1979	0.0	0.0	0.0
1980-1989	0.0	0.0	0.0
1990-1999	0.0	0.0	0.0
2000-2009	0.0	0.0	0.0

Fig. 1: An example of a confusing image, with the ground truth labelled with the red rectangle.

Hence, heuristics methods that cannot adapt to the various types of table formats would not be able to detect tables whose format differs significantly from conventional tables (as seen in Figure 1). Thus, machine learning is apt for training algorithms that can detect tables as it can account for the large variety of possible formats.

An inherent limitation of this project is the limited amount of training data available. To keep the run time and RAM used feasible, limited computational resources are used, and this restricted the complexity of our model architecture.

## 2 AIMS/OBJECTIVES

We aim to perform fully automatic detection of borderless tables with a higher accuracy and precision. This can be done with deep learning algorithms with transfer learning. With an accurate and precise localisation of the tables, this could aid the next steps in analysing and extracting the contents of the table in the document image.

Moreover, we also aim to investigate the effect of various kernel sizes on a table detection model's performance.

In addition, we would also like to investigate the performance of two-stage detectors compared to one-stage detectors trained via deep learning in detecting borderless tables.

## 3 LITERATURE REVIEW

As identified by previous works such as DeepDeSRT<sup>[3]</sup>, TableNet<sup>[4]</sup>, and in a paper done by Adam et. al<sup>[5]</sup>, the task of table detection is indeed similar to the task of general object detection. This refers to the task

of detecting where the objects are, and types of objects detected in general images. It would therefore be possible to employ deep learning techniques used with general object detection on the task of table detection<sup>[6]</sup>, including the detection of borderless tables.

In DeepDeSRT, Ren et al. proposed the use of Faster RCNN<sup>[7]</sup> model with a VGG-16<sup>[8]</sup> backbone to detect tables from document images. While this approach was one of the first table detection methods based on deep learning and object detection, their model can be further customised on the task of table detection, where the images tend to be simpler and the features tend to be rectangular.

In RetinaNet<sup>[9]</sup>, Lin et. al proposes a one-stage detector whose novelty is its loss function that reduces class imbalances to improve its accuracy. According to Lin et. al, one-stage detectors tend to have lower accuracy due to the need to process a much larger set of candidate object locations regularly sampled across an image, which leads to a more significant class imbalance from easy negatives. With its unique loss function, RetinaNet is able to prevent these easy negatives from overwhelming the one-stage detector, and have an accuracy on-par with two-stage detectors such as Faster RCNN with a training speed on-par with one-stage detectors such as YOLOv2<sup>[10]</sup> and DSSD513<sup>[11]</sup>.

## **4 EXPERIMENTS AND METHODOLOGY**

### **4.1 Type of detector**

We chose to modify Faster RCNN over RetinaNet, as Faster RCNN is a two-stage detector, which generally has a higher accuracy at the expense of slower training speed. Meanwhile, RetinaNet is a one-stage detector, which is generally faster but has a lower accuracy. Training speed is not as relevant for table identification as it does not affect the table identification process after the training of weights is complete. Hence, we chose to prioritize accuracy, as missing out parts of a table can have significant implications on the amount of information that can be successfully extracted.

Although RetinaNet has slightly better average precision compared to Faster RCNN, so does DeepDeSRT, so we chose to improvise Faster RCNN by taking inspiration from DeepDeSRT while maintaining it as a two-stage detector. However, as RetinaNet's performance is relatively good compared to other one-stage detectors, it can act as a good benchmark for one-stage detectors in general.

### **4.2 Model architecture**

Taking inspiration from DeepDeSRT, we implemented Faster RCNN with a VGG-16 backbone. For the purpose of comparison, we also trained a RetinaNet model with a Resnet-50 backbone<sup>[12]</sup> on the same data, through the same process.

In addition, in the process of detecting tables in document images, we notice that many of the low-level features, such as contiguous sections of blank spaces in the borderless table margins, are all rectangular. Therefore, we also employed the use of rectangular convolutions in the first 2 layers of the backbone. We experimented with various kernel sizes (of  $L \times 3$  where  $L$  is an odd integer we varied). We added suitable padding to ensure both kernel sizes would produce feature maps of the same shape.

Since some of these features are vertical and others are horizontal (given that tables can be both horizontal and vertical), we used both horizontal and vertical convolutions (of shapes  $L \times 3$  and  $3 \times L$ ), equally distributing them among the 64 feature maps.

### 4.3 Dataset

To train our network to detect tables, we used the document images from the ICDAR 2013<sup>[13]</sup>, ICDAR 2019<sup>[14]</sup>, and the UNLV table dataset<sup>[15]</sup>, to form our combined dataset of 1954 images. In order to fine-tune the network on borderless tables, we also used an online dataset<sup>[16]</sup> which contained only borderless or semi-bordered tables. As for evaluation, we used the same online dataset's test set of 65 document images, containing 100 borderless or semi-bordered tables.

Table 1: Details about the datasets we used for training our model

Dataset	ICDAR 2013	ICDAR 2019	UNLV	Online dataset
Number of images	78	1200	338	338
Number of tables	98	1656	418	1254

### 4.4 Training specifications

All experiments were run on Google Colaboratory platform's cloud GPU with 16 GB of GPU memory, and Intel(R) Xeon(R) CPU @ 2.20GHz and 12.72 GB of RAM. All networks were trained using Python 3.6.9. Faster RCNN was trained using Tensorflow version 2.3.0, while RetinaNet was trained using PyTorch version 1.7.0.

To train all of the networks, we used SGD optimiser from both Keras and PyTorch with a fixed learning rate of 0.0001 and a batch size of one as suggested by Ren et al.<sup>[7]</sup>

### 4.5 Image augmentation

In order to improve results, we used image augmentation to expand the training dataset. Providing more training data, especially cases which are more difficult, is crucial to produce deep learning models that

perform better, as well as help prevent overfitting where the model may do poorly. However, document images are generally axis-aligned. Therefore, some traditional image augmentation methods (such as rotation or shearing) may not be useful with document images as they will affect the image's axis-alignment.

Therefore, we used right-angle rotations, as well as vertical and horizontal flips, since these will ensure the tables remain axis-aligned. In addition, we used the smudge and dilation transformations as proposed by Devashish Prasad et al.<sup>[17]</sup> These transformations would produce different document images which have the same table locations. Examples of dilated and smudged images are shown in Figure 2.

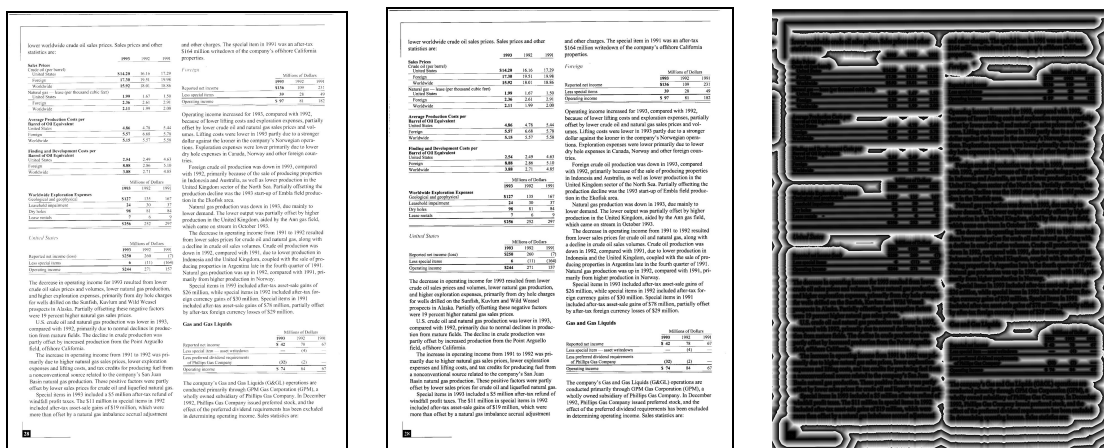


Fig. 2: Original image from the online dataset, its dilated version, and its smudged version, from left to right

## 4.6 Iterative Transfer Learning

Huge amounts of training data are imperative for deep learning object detection models. Thus, in addition to image augmentation in 4.5, we also used iterative transfer learning, where knowledge gained from solving one problem is used to solve another problem. This would allow for faster convergence of the models as well as improved performance.<sup>[18]</sup>

To train our networks, we firstly initialise the models with the weights of the model pre-trained on general object detection, by Ren et al. Due to the differences in model architecture as well as the differing number of object classes, for Faster RCNN, we only did this with the backbone network excluding the layers we modified.

Next, we train this initialised model on general table detection for 80 epochs. As having a larger training sample generally improves the models' performance, we trained our models on our combined dataset of 1954 document images. This will better fine-tune the backbone network to pick out features in document images. At the same time this allows the general network to learn to detect tables based on higher-level features such as the systematic arrangement of table cells.

Lastly, we train our models on detection of borderless tables for another 80 epochs. We did not freeze the weights of the backbone network to allow it to better capture the features of borderless tables. This training is done with the smaller online dataset of borderless tables.

As seen from Figure 3.1, training 80 epochs was indeed sufficient for the Faster RCNN model to converge initially as the loss stabilised. However, the next training phase evidently helped to further improve the model as seen from the further reduced loss, stabilising after about 140 epochs. This shows that our choice of training it for 160 epochs was indeed enough.

Similarly, as seen from Figure 3.2, the loss has stabilised after 80 epochs for RetinaNet. The spike after 80 epochs is likely caused by the switch from the combined dataset to the dataset containing only borderless tables. The RetinaNet model may have a harder time detecting borderless tables, resulting in the spike. However, the loss soon stabilised after about 140 epochs, so our choice of training it for 160 epochs was enough too.

Note that both models use different loss functions.

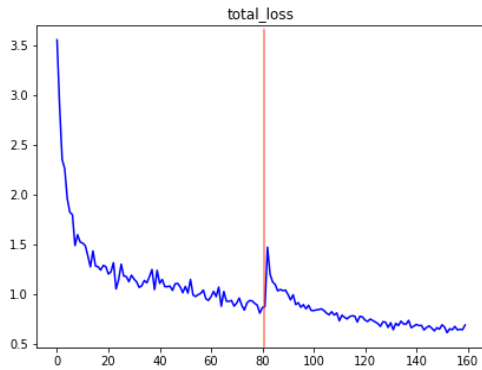


Fig. 3.1: The total loss of each epoch over 160 epochs for Faster RCNN

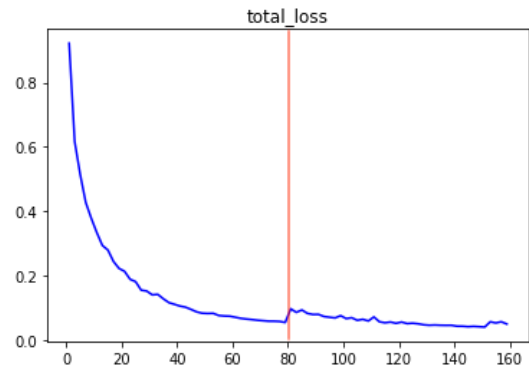


Fig. 3.2: The total loss of each epoch over 160 epochs for RetinaNet

## 5 RESULTS

To evaluate our model's performance, we tested our model on the test dataset from the online dataset we used containing 65 images and 100 borderless and semi-bordered tables. We then calculated its recall, precision and F1-measure results<sup>1</sup> (weighted average of recall and precision) for Intersection over Union (IoU)<sup>2</sup> values of 0.5 to 0.8, as seen in Table 2.

<sup>1</sup> For explanation of precision, recall, and F1-measure refer to Appendix 1.

<sup>2</sup> For explanation of IoU, refer to Appendix 2.

Table 2: Borderless table detection results for Faster RCNN and RetinaNet

	Recall		Precision		F1-measure	
IoU	Faster RCNN	RetinaNet	Faster RCNN	RetinaNet	Faster RCNN	RetinaNet
0.5	<b>0.82</b>	0.45	0.79	<b>0.98</b>	<b>0.80</b>	0.62
0.6	<b>0.79</b>	0.41	0.76	<b>0.96</b>	<b>0.77</b>	0.60
0.7	<b>0.67</b>	0.43	0.65	<b>0.95</b>	<b>0.66</b>	0.59
0.8	<b>0.46</b>	0.41	0.44	<b>0.90</b>	0.45	<b>0.56</b>

Compared to Faster RCNN, RetinaNet has a very high precision, but very low recall where less than half of the tables are detected even at low IoU thresholds. When comparing their F1-measures, Faster RCNN has a generally higher F1-measure than RetinaNet, with the exception of IoU equals 0.8.

We also trained a few models while varying the kernel size of the rectangular convolutional filters, of shapes 3 x 3 (no modifications to Faster RCNN), 5 x 3, 7 x 3, 9 x 3, 11 x 3, and 13 x 3, while adding suitable padding of 1 x 1, 2 x 1, 3 x 1, 4 x 1, 5 x 1 and 6 x 1 respectively. As such models take a long time to be trained and the results in table 2 showed that Faster RCNN would yield more promising results, we focused on Faster RCNN in this stage. The F1-measures are shown below in Table 3.<sup>3</sup> As seen in Table 3, there is a generally increasing trend in the performance of the model as the size of the rectangular convolution kernels increases. This is indeed expected, as increasing the size of the kernels will increase the number of parameters in the model, which theoretically allows the model to perform better.

Table 3: F1-measure of our model with various kernel sizes for different IoU

	Kernel size					
IoU	3x3	5x3	7x3	9x3	11x3	13x3
0.5	0.80	0.88	0.83	0.84	<b>0.86</b>	0.84
0.6	0.77	0.77	0.75	0.79	<b>0.79</b>	0.79
0.7	0.66	0.62	0.64	0.68	<b>0.68</b>	0.70
0.8	0.45	0.44	0.46	0.47	<b>0.49</b>	0.46

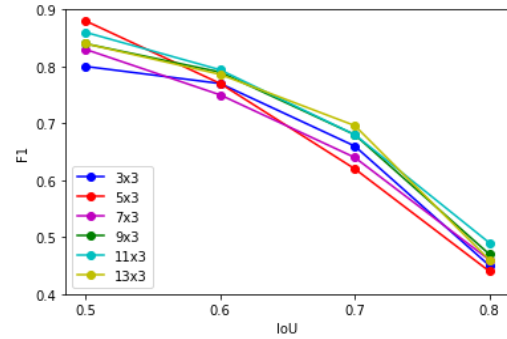


Fig. 4: F1-measure with various kernel sizes for different IoU

However, as the kernel size increases from 11 x 3 to 13 x 3, the performance does not increase for most IoU thresholds, and even decreases with IoU thresholds of 0.5 and 0.8. This is likely due to overfitting, where the model becomes overly complex with too many trainable parameters. This would cause it to fit

<sup>3</sup> For recall and precision of different kernel sizes, refer to Appendix 3.

too closely on the training data and fail to generalise on images outside the training data. Based on this data, we decided to use rectangular convolutions of shape 11 by 3, where we generally attain the best performance and it still shows little signs of overfitting. The F1-measures from Table 3 are illustrated in Figure 4, which show that rectangular convolutions of shape 11 by 3 are indeed optimal.

## 6 DISCUSSION

### 6.1 Results Analysis

As observed from Figure 5, our model is indeed able to detect tables of a wide variety of sizes. It also works for documents of different types, varying from simpler documents with just a table to more difficult cases where smaller tables are embedded into the text, which is harder to distinguish.

This shows that our model is able to handle a wide range of cases, successfully handling tables embedded into documents in different ways. It also performs well with intra-class variability as many of these tables are drawn in different ways.

Figure 5 shows three sample documents with detected tables highlighted by red boxes. The first document is a 'Statement of Consolidated Shareholders' Equity' with a table of financial data. The second document is a 'Financial Statement' with a table of financial data. The third document is a 'Financial Statement' with a table of financial data.

Fig. 5: Our model's table detection results

### 6.2 Error Analysis

However, there are also some cases where our model may not perform well, as seen in Figure 6.

Figure 6 shows two sample documents where the model's performance is poor. (a) Merging close tables: A document with two tables that are too close together, and the model has merged them into a single table. (b) Misidentified tables: A document with a table that the model has misidentified as a different type of table.

(a) Merging close tables

(b) Misidentified tables

Fig. 6: Cases where our model does not perform well



For example, in cases where a few tables are close together (Figure 6a), our model may face difficulties in distinguishing them. This may be due to a lack of training data with similar cases of close tables.

Further qualitative analysis in Figure 6b shows that the model may indeed misinterpret the margins in the document to be features of a table. This then results in false positives that were identified wrongly as tables. However, this is an inherent difficulty with detecting borderless tables, where the document margins may be confusing.

### 6.3 Faster RCNN vs RetinaNet

As the results in Table 2 have shown, RetinaNet has a very high precision, but a very low recall value compared to Faster RCNN.

This makes RetinaNet not as suited for table detection. A lower precision indicates that more computational power and time will be wasted on detecting the wrong tables and extracting unnecessary information. However, a low recall value indicates that more tables are not detected by the detection algorithm, which may result in the loss of vital information. As having additional unnecessary information is usually more preferable than risking essential information being lost, Faster RCNN is more suited for table detection than RetinaNet with a higher recall, albeit with a lower precision. The consistently higher F1 measure by Faster RCNN compared to RetinaNet in Table 2 further solidifies this conclusion.

### 6.4 Application

Additionally, we attempted to apply our model on a range of real-world examples outside of just typical document images. These include a range of receipts, bus timetables as well as parking rate tables as seen in Figure 7a and 7b.

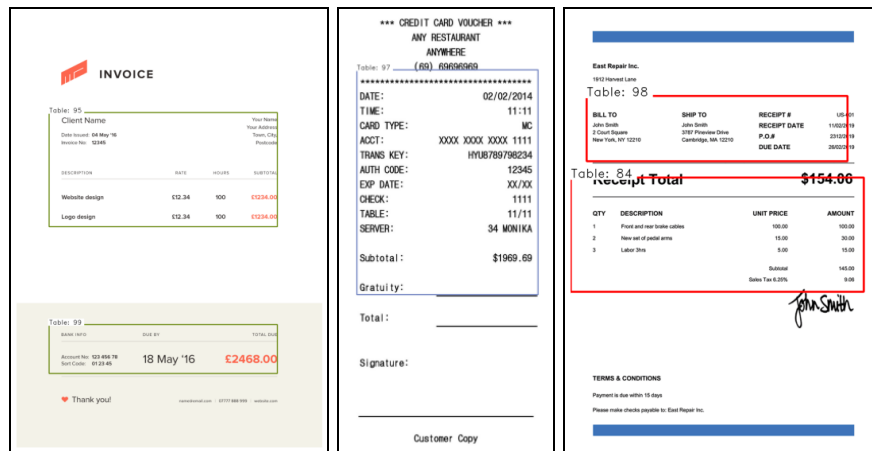


Fig. 7a: Our model detects tables in receipts where information is contained in tables or table-like structures accurately



## REFERENCES

- [1] Bo-Christer Bjork, A. (n.d.). Scientific journal publishing: Yearly volume and open access availability. Retrieved January 1, 2021, from <http://informationr.net/ir/14-1/paper391.html>
- [2] Bavdekar, Sandeep. (2015). Using Tables and Graphs for Reporting Data. The Journal of the Association of Physicians of India. 63. 59.
- [3] Schreiber, Sebastian & Agne, Stefan & Wolf, Ivo & Dengel, Andreas & Ahmed, Sheraz. (2017). DeepDeSRT: Deep Learning for Detection and Structure Recognition of Tables in Document Images. 1162-1167. 10.1109/ICDAR.2017.192.
- [4] Paliwal, Shubham & D, Vishwanath & Rahul, Rohit & Sharma, Monika & Vig, Lovekesh. (2019). TableNet: Deep Learning Model for End-to-end Table Detection and Tabular Data Extraction from Scanned Document Images. 10.1109/ICDAR.2019.00029.
- [5] Harley, Adam & Ufkes, Alex & Derpanis, Konstantinos. (2015). Evaluation of deep convolutional nets for document image classification and retrieval. 991-995. 10.1109/ICDAR.2015.7333910.
- [6] Gilani, Azka & Qasim, Shah Rukh & Malik, Imran & Shafait, Faisal. (2017). Table Detection Using Deep Learning. 10.1109/ICDAR.2017.131.
- [7] Ren, S., He, K., Girshick, R., & Sun, J. (2016, January 06). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. Retrieved January 25, 2021, from <https://arxiv.org/abs/1506.01497>
- [8] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), 2015.
- [9] Lin, Tsung-Yi & Goyal, Priya & Girshick, Ross & He, Kaiming & Dollar, Piotr. (2017). Focal Loss for Dense Object Detection. 2999-3007. 10.1109/ICCV.2017.324.
- [10] J. Redmon and A. Farhadi. YOLO9000: Better, faster, stronger. In CVPR, 2017.
- [11] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. DSSD: Deconvolutional single shot detector. arXiv:1701.06659, 2016.

- [12] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In CVPR, 2017.
- [13] Gobel, M., Hassan, T., Oro, E., & Orsi, G. (n.d.). ICDAR 2013 Table Competition. Retrieved May 19, 2020, from <http://www.orsigiorgio.net/wp-content/papercite-data/pdf/gho13.pdf>
- [14] L. Gao et al., "ICDAR 2019 Competition on Table Detection and Recognition (cTDaR)," 2019 International Conference on Document Analysis and Recognition (ICDAR), Sydney, Australia, 2019, pp. 1510-1515, doi: 10.1109/ICDAR.2019.00243.
- [15] Shahab, Asif & Shafait, Faisal & Kieninger, Thomas & Dengel, Andreas. (2010). An open approach towards the benchmarking of table structure recognition systems. ACM International Conference Proceeding Series. 113-120. 10.1145/1815330.1815345.
- [16] Sgrpanchal31. (n.d.). Sgrpanchal31/table-detection-dataset. Retrieved June 15, 2020, from <https://github.com/sgrpanchal31/table-detection-dataset>
- [17] Prasad, Devashish & Gadpal, Ayan & Kapadni, Kshitij & Visave, Manish & Sultanpure, Kavita. (2020). CascadeTabNet: An approach for end to end table detection and structure recognition from image-based documents.
- [18] Weiss, K., Khoshgoftaar, T.M. & Wang, D. A survey of transfer learning. J Big Data 3, 9 (2016). <https://doi.org/10.1186/s40537-016-0043-6>

## Appendix 1

Precision, recall and F1-measure are metrics that measure the accuracy of predictions. Precision quantifies the number of true positive predictions made out of all the predictions, recall quantifies the number of true positive predictions made out of all possible positive predictions, while F1-measure is a weighted average of precision and recall.

The precision value is calculated by  $\frac{TP}{TP + FP}$ , where TP represents the number of true positives, and FP represents the number of false positives.

On the other hand, the recall value is calculated by  $\frac{TP}{TP + FN}$ , where TP represents the number of true positives, and FN represents the number of false negatives.

True positives refer to predictions where the IoU<sup>4</sup> between the prediction bounding boxes and the ground-truth bounding boxes are above the IoU threshold. False positives refers to predictions where the IoU between the prediction bounding boxes and the ground-truth bounding boxes are below the IoU threshold. False negatives refer to predictions where the bounding boxes are not identified.

F1-measure is given by the formula  $\frac{2PR}{P+R}$ , where P is the precision and R is the recall, to give a weighted average of precision and recall.

---

<sup>4</sup> For explanation about IoU, refer to Appendix 2.

## Appendix 2

Intersection over Union is an evaluation metric used to measure the accuracy of an object detector on a particular dataset. Any algorithm that provides predicted bounding boxes as output can be evaluated using IoU.

Two items are needed to calculate the IoU:

1. The ground-truth bounding boxes (i.e., the bounding boxes that specify where in the image our object is).
2. The predicted bounding boxes from our model.

Table 5-3  
Comparison of Reference-Case First-Repository Costs  
for the Improved-Performance System with the 1986 Estimates  
(Billions of dollars)

Repository Site	Construction	Operation	Closure and Decommissioning	Total
<b>Basalt</b>				
1987	3.04	6.89	0.51	10.44
1986	1.84	8.15	-0.22	10.21
Change <sup>a</sup>	+1.20	-1.26	+0.29	+0.23
<b>Salt</b>				
1987	1.78	5.80	0.35	7.93
1986	1.60	5.09	-0.19	6.88
Change <sup>a</sup>	+0.18	+0.71	+0.16	+1.05
<b>Tuff</b>				
1987	0.84	4.21	0.37	5.42
1986	0.98	4.30	-0.07	5.35
Change <sup>a</sup>	-0.14	+0.09	+0.30	+0.07

<sup>a</sup>Estimates from U.S. Department of Energy, Office of Civilian  
Radioactive Waste Management, Analysis of the Total System Life Cycle Cost for  
the Civilian Radioactive Waste Management Program, DOE/RW-0047, April 1986.  
<sup>b</sup>The 1987 estimate minus the 1986 estimate.

-87-

Fig. 8: An illustration of the two bounding boxes.

In Figure 8, the green box is the ground-truth bounding box that specifies the position on the car in the image, while the red box is the predicted bounding box from a model.

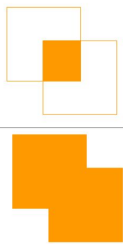
$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


Fig. 9: A visual illustration of IoU

To calculate the IoU of a prediction, the area of overlap between the two bounding boxes is divided by the area of union of the two bounding boxes.

Hence, the closer the two bounding boxes are, the higher the IoU, and hence the better the predictions.

## Appendix 3

Table 4 and 5 shows the recall and precision of our model respectively with various kernel sizes for different IoU. These values are used to compute the values shown in Table 3.

Table 4: Recall of our model with various kernel sizes for different IoU

	Kernel Size					
IoU	3x3	5x3	7x3	9x3	11x3	13x3
0.5	0.82	0.89	0.86	0.89	0.88	0.84
0.6	0.79	0.77	0.77	0.86	0.81	0.79
0.7	0.67	0.63	0.66	0.75	0.69	0.7
0.8	0.46	0.45	0.48	0.53	0.5	0.46

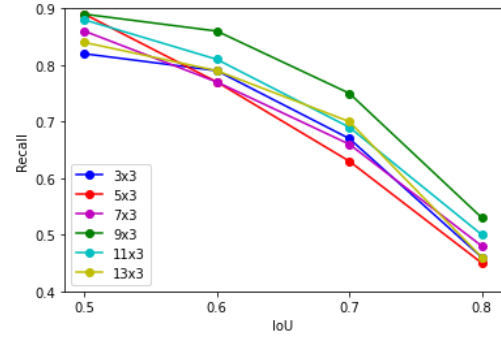


Fig. 10: Recall of our model with various kernel sizes for different IoU

Table 5: Precision of our model with various kernel sizes for different IoU

	Kernel Size					
IoU	3x3	5x3	7x3	9x3	11x3	13x3
0.5	0.80	0.87	0.82	0.80	0.85	0.83
0.6	0.77	0.75	0.73	0.74	0.78	0.78
0.7	0.65	0.62	0.63	0.64	0.66	0.69
0.8	0.44	0.44	0.46	0.43	0.48	0.46

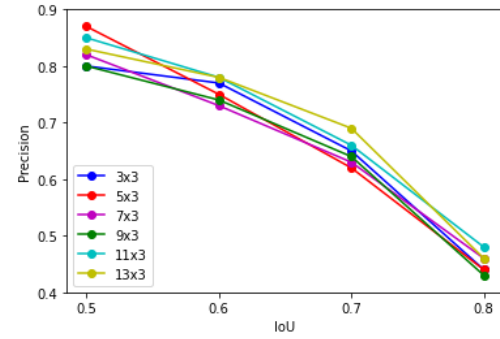


Fig. 11: Precision of our model with various kernel sizes or different IoU