Error Analysis of Object Detection Models With Interactive User Interfaces

by
Dayeon Oh

A Final Project

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented June 9th
Commencement 2022

AN ABSTRACT OF THE PROJECT OF

Dayeon Oh for the degree of Master of Science in Computer science presented on June 9th, 2022.

Title: Error Analysis of Object Detection Models with Interactive User Interfaces.

Object detection models are being widely used in many applications, such as autonomous driving, construction management, and cancer detection. Evaluating the performance of the object detection model is more complicated than other computer vision models such as image classification models. Most of the images have several objects to be detected, and the types of detections and their errors can be categorized in several different ways (e.g., classification error, localization error). To address this challenge, we design and develop an interactive tool that helps users evaluate and analyze results from object detection models. We first categorize detected and ground-truth bounding boxes into 7 and 11 types based on the literature. We enable users to analyze object detection results based on this categorization at three different levels: summary level, class and detection type level, and image level. This enables users to analyze a large number of and a variety of model errors from a summarized overview into individual images. From the summary level, users can explore the overall detection results such as average precision, the number of detected labels and ground truth labels, and the number of each detection type. At the class and detection level, users can see more detailed information about a certain class and detection type and images that correspond to it. At the image level, users can click each image to get a detailed analysis. We developed an interactive user interface that implements this idea with a driving dataset, trained with a state-of-the-art object detection model. We also present a usage scenario of how users can use our tool for inspecting errors from the trained object detection model. Being able to easily browse object detection results and their images with a certain class and a detection type, our interface helps users to work with object detection models more effectively in their research.

Corresponding e-mail address: ohda@oregonstate.edu

TABLE OF CONTENTS

Page

1 Introduction 1

2 Related works...4

3 Methods and Design 7

  3.1 Tool development..7

  3.1.1 Summary view panel 7

    3.1.2. Chart view panel..14

    3.1.3 Image view panel 17

  3.2 Post-processing 18

4 Use cases...21

  4.1 Dataset and Model...21

  4.2 User scenarios 21

  4.2.1 Finding similarities between certain errors 22

  4.2.2 Finding example images for specific conditions...23

5 Discussion and Future work...25

Bibliography 26

LIST OF FIGURES

Figure Page

1. The object detection error explorer. 6
2. Ground truth result summary table from the summary view panel 6
3. Summary view panel 8
4. Detected label types 10
5. Ground-truth label types 13
6. Image view panel 14
7. Image modal window panel 16
8. Chart images for user scenarios 19

Error Analysis of Object Detection Models with Interactive User Interfaces

1 Introduction

Recent advancements in deep learning and hardware computing power have increased the capabilities of computer vision (CV), a scientific field that defines how machines understand the meaning of images and videos like humans do. In general, CV tasks consist of image classification, object detection, facial recognition, and image segmentation. Among the CV tasks above, object detection, a task that works to identify and locate objects within an image or video has been one of the successful applications of CV in many fields such as robotics [8], transportation (e.g., autonomous driving [5]), and even in a medical [6] and construction area [7].

To use the object detection model more confidently, the interpretability of a model with its performance is particularly important because the end-users can assess how well the model performs on a specific object detection task. Moreover, the users can make more critical decisions based on the results of the object detection model only when they interpret and rely on the model. Also, if the model is interpretable enough, the end-users readily understand the reasoning behind the predictions and decisions made by the model even without domain knowledge of computer science. The interpretability of the model can be improved when end-users recognize and correct predictions (i.e., object detection) that are incorrect. One key metric of the improved interpretability is its error debugging and intuitive diagnosis for the model’s detections and predictions. The users may need to debug any errors made from the model by evaluating and analyzing its prediction performance together with quantitative result tables of the model.

Evaluating the performance of the object detection model is particularly challenging compared to other CV tasks such as image classification. It is because the object detection results can be categorized into many types, not just as correct and incorrect unlike other computer vision models such as image classification models. In further, several detected bounding boxes can be detected correctly but their detected class can be incorrect, and several bounding boxes are detected incorrectly but their detected class can be correct. As a result, an effective performance evaluation method for object detection is needed in our body of knowledge. Meanwhile, to evaluate the performance of the object detection model, we may use various evaluation metrics such as mean average precision (mAP) and harmonic mean of precision and recall (F1-score), but they are not explaining detailed information about the errors.

Therefore, as mentioned above we can specify the detection errors into sub-categories to understand the errors of detection results in more detail. In addition, sorting out images that contain particular labels, classes, or error types, and analyzing errors can be more complicated compared to other computer vision models. This issue is particularly important for non-computer science experts because it would be much harder for them to understand the model's performance with their limited expertise and experiences.

Previous research suggested various solution tools such as Voxel 51 [4], TIDE [1], and Hoiem et al [2] to address the problems above. Their framework provides the interface to look for the detection result of each image or provide chart information of the errors. Also, these works have categorized the incorrect detection results into several different groups to provide users with error information that was difficult to understand with mAP or correct and incorrect results alone. However, previous research works still have limitations as follows:

1. Focusing only on showing bounding boxes of each image or summarizing detection errors causes a lack of information about the opposite.
2. Categorized error types do not include all the detected errors.

This paper presents the object detection error analyzer [Figure 1], which is an interpretability tool that supports the evaluation and analysis of the result of the object detection model. Our tool will help users to work with the object detection model more efficiently by providing three different level approaches to the object detection results, which are summary level, class and detection type level, and image level. We designed it into three different levels to provide adequate information, from the overall to detailed results of the model, to match the users' needs. Also, we expanded six different detection errors created from the TIDE [1] into seven different categories including correctly classified bounding boxes and other remaining incorrect bounding boxes which were not included in the former six detection errors. In addition, we also categorized the ground truth bounding based on the detected bounding boxes and their detection status to show a relationship between detected and ground-truth bounding boxes.

First, at the summary level, we designed a table and distribution charts that contain the overall and summarized results of the object detection model. From the table, users will be able to see basic information such as numbers and detected and ground truth bounding boxes, average precisions, and the number of labels belonging to each detection type based on the classes. Users can also look for the distribution of the size of bounding boxes, IOU, and detection score of all detected labels through distribution charts. Second, in the class and detection type level, based on the types that we classified the detection types and classes users can look for the corresponding images, distribution chart results of bounding boxes, IOU, and detection score. Third, at the image level, we designed a format that users can analyze all the detected and ground truth bounding boxes to get more detailed information about each selected image.

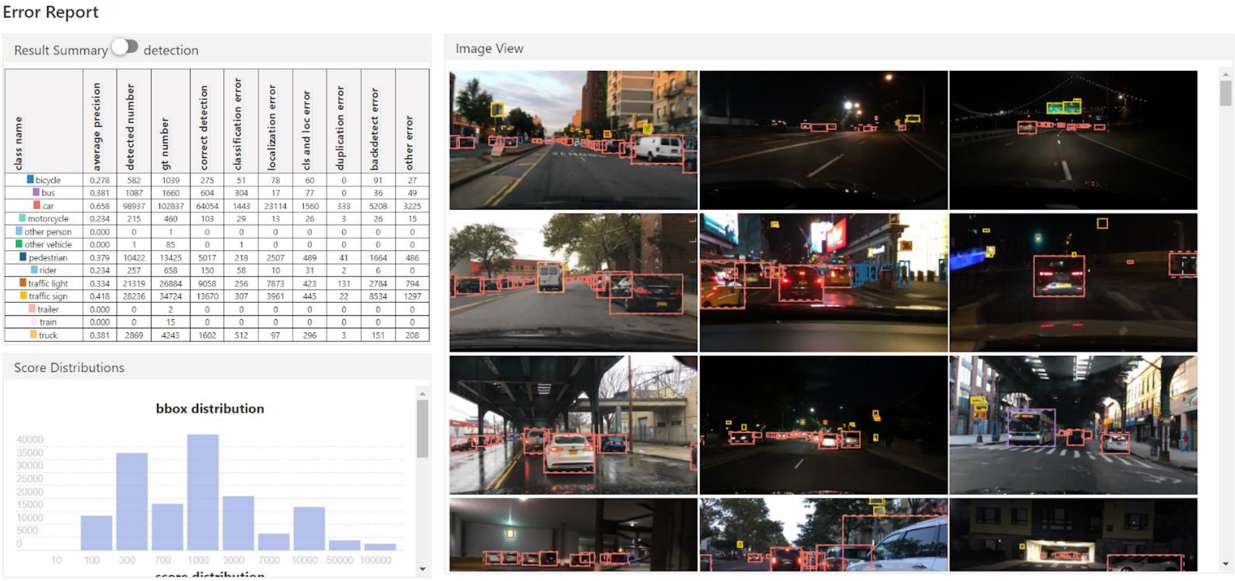
With these features, this tool can support end-users to check the performance of the object detection model from overall summary to small detail. Also, it can help them to make better decisions for their purposes with reliability and accountability by effectively combining interpretability in the object detection model. Furthermore, from a technical point of view, this tool provides visualization of the performance and presents a systematic analysis method. This paper will first describe the dataset, models, and post-processing methods, and then will describe the tool of design, and walk through some user scenarios.

2 Related Works

Academy-wise, mAP score has been traditionally used as an indicator to measure the performance of object detection models, despite its drawback that it does not provide insights into the errors. Therefore, many researchers have made efforts to come up with better tools to measure the performance of the object detection models, especially in many cases by providing analytical information about the errors. Hoeim et al [2], and TIDE [1], created their own labels to categorize the object detection errors. Hoeim et al [2], showed which aspects influence the object detection and categorized the false-positive errors into four different categories: localization error, confusion with similar objects, confusion with dissimilar objects, and confusion with the background. They showed the results of the false-positive errors of each class in the form of a pie chart and a bar chart. TIDE [1] categorized the errors into 6 different categories: classification error, localization error, classification and localization error, duplication error, background error, and missed detected error. They also used pie charts and bar charts to show the distribution of errors in each class and showed charts for different object detection models to compare performances. They compared different models and different datasets to evaluate the performance. In contrast, we have extended their 6 different error categories to categorize the object detection results.

Borji et al [3], introduced a new evaluation method called upper bound average precision (UAP). They calculated the UAP of each class, and also compared about 15 different object detection models, and evaluate and visualize the performance with graphs and tables. They showed the performance of different models for each class by using different icons and displaying them in the same bar, using the y-coordinate to illustrate the performance. Google open image dataset [12] created a browsing tool for images. It is not specific for the object detection models, but users can select filters such as class, train, or validation data to browse corresponding images. It allows users to browse individual images with objects and their bounding boxes and classes. Voxel 51 [4] introduced an interactive tool to explore and browse the object detection model. They helped users to investigate the object detection results with the filtering functions such as labels, categories, and detection scores. It is possible to create a summary result that shows precision, recall, f-1 score, and support for each class. However, it has a shortcoming in that users have to write a python code to use this tool.

Hoeim et al [2], TIDE [1], and Borji et al [3] showed the performance of object detection models by summarizing errors into the types they have created. Voxel 51 [4] and Google open image dataset [12] displayed the images that users have filtered. When evaluating the performance of object detection models and debugging them, it is important for users to intuitively understand how the objects were detected when they look at the results. Because of this reason, many researchers have tried to categorize and show errors with simple and compact visualizations. However, 4 different false-positive error types and 6 error types that previous research has created do not contain all the detection results which limit the explanation of the results. In addition, showing summary results with pie charts and bar charts is easy to check the percentage of each error type, but it is difficult to immediately check in detail which images and under what circumstances each error occurs. In this regard, we create an interactive tool that displays a summary, charts, and images all together to provide a simultaneous exploration of the results with example images. Also, we create new categories that include all the detected errors to allow users to browse all the objects and images based on the selected category. In this way, users can recognize the similarities among the objects that belong to the same category and check the overall performance of object detection models.

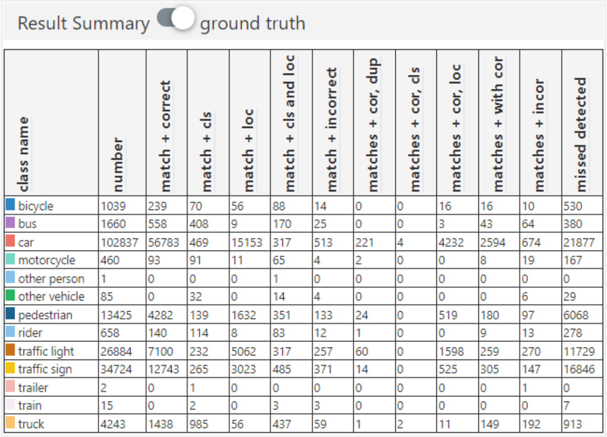


Error Report

Result Summary detection

class name	average precision	detected number	gt number	correct detection	classification error	localization error	cls and loc error	duplication error	backdetect error	other error
■ bicycle	0.278	582	1039	275	51	78	60	0	91	27
■ bus	0.381	1087	1660	604	304	17	77	0	36	49
■ car	0.658	98937	102837	64054	1443	23114	1560	333	5208	3225
■ motorcycle	0.234	215	460	103	29	13	26	3	26	15
■ other person	0.000	0	1	0	0	0	0	0	0	0
■ other vehicle	0.000	1	85	0	1	0	0	0	0	0
■ pedestrian	0.379	10422	13425	5017	218	2507	489	41	1664	486
■ rider	0.234	257	658	150	58	10	31	2	6	0
■ traffic light	0.334	21319	26884	9058	256	7873	423	131	2784	794
■ traffic sign	0.418	28236	34724	13670	307	3961	445	22	8534	1297
■ trailer	0.000	0	2	0	0	0	0	0	0	0
■ train	0.000	0	15	0	0	0	0	0	0	0
■ truck	0.381	2869	4243	1602	512	97	296	3	151	208

Figure 1. The Object Detection Error Explorer. Summary view panel (A), Chart view panel (B), and Image view panel (C).



Result Summary ground truth

class name	number	match + correct	match + cls	match + loc	match + cls and loc	match + incorrect	matches + cor, dup	matches + cor, cls	matches + cor, loc	matches + with cor	matches + incor	missed detected
■ bicycle	1039	239	70	56	88	14	0	0	16	16	10	530
■ bus	1660	558	408	9	170	25	0	0	3	43	64	380
■ car	102837	56783	469	15153	317	513	221	4	4232	2594	674	21877
■ motorcycle	460	93	91	11	65	4	2	0	0	8	19	167
■ other person	1	0	0	0	1	0	0	0	0	0	0	0
■ other vehicle	85	0	32	0	14	4	0	0	0	0	6	29
■ pedestrian	13425	4282	139	1632	351	133	24	0	519	180	97	6068
■ rider	658	140	114	8	83	12	1	0	0	9	13	278
■ traffic light	26884	7100	232	5062	317	257	60	0	1598	259	270	11729
■ traffic sign	34724	12743	265	3023	485	371	14	0	525	305	147	16846
■ trailer	2	0	1	0	0	0	0	0	0	0	1	0
■ train	15	0	2	0	3	3	0	0	0	0	0	7
■ truck	4243	1438	985	56	437	59	1	2	11	149	192	913

Figure 2. Ground truth result summary table from the summary view panel.

3 Methods

The object detection result analyzer is an interactive tool that supports the evaluation and analysis of the results of the object detection model. The objective of our research is to develop a tool that enables debugging object detection models and verifies the features of the tool. For this purpose, we design the tool as a one-page web application with the features to analyze and evaluate the object detection results. In addition, we use a realistic user scenario to verify the function of the tool. We design three different panels to debug the object detection results by categorizing the object detection ground truths and detection labels, displaying distribution charts of different values such as IoU, detection score, and bounding box size. Finally, we display images with the bounding boxes to let users explore and interpret the object detection results easily.

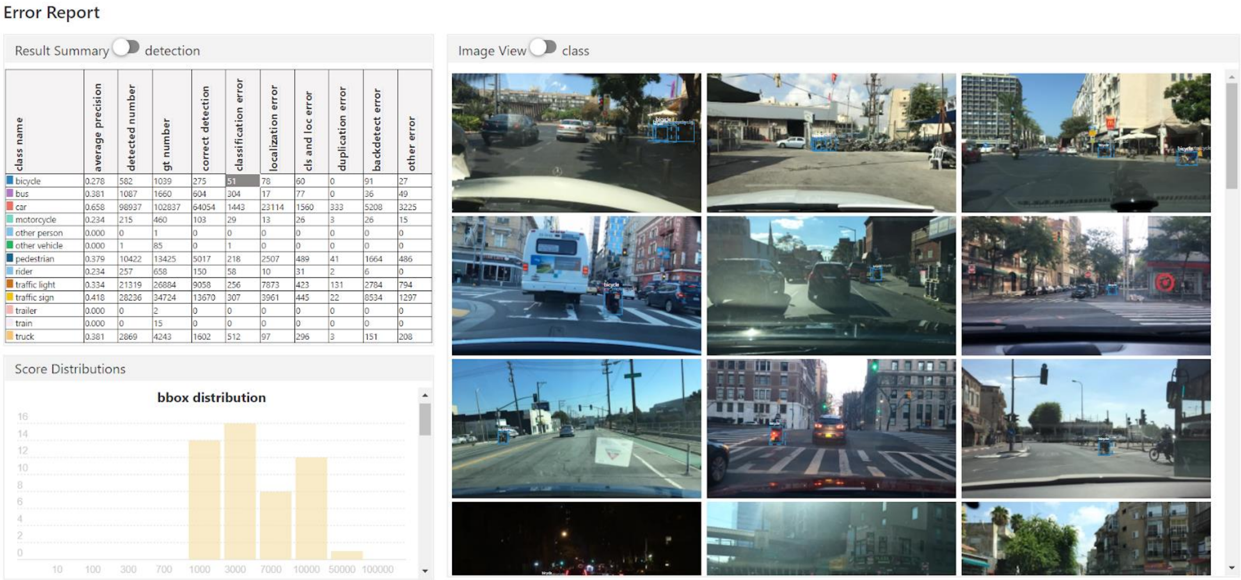
3.1 Tool development

This section describes how we design the tool to implement the features to accomplish our research goal. Object detection result explorer is a one-page web application that provides three different views to analyze the object detection results: summary level table view, chart view, and image view.

3.1.1 Summary view panel

We design a summary result table [Figure 1. A] to display basic information about the object detection results. It contains the number of detected labels and ground truth labels of each class and their average precision values. We design a detected result summary table and ground truth summary table to check how many labels are in each class at a glance. One of the main information is the number of detected labels of each detected category by each class. We categorize the detection result and ground truth values in several different categories based on the literature so that end-users can easily explore the object detection results of their interests.

The most frequently used evaluation metric for the object detection model is the mean average precision (mAP). This measurement can show the accuracy of the model's performance with a simple numeric number. However, it is difficult to conduct a detailed analysis of the object detection model's results with this single result alone because it does not show any information about the incorrectly detected labels. Therefore, we categorize the detected labels into 7 different categories. In addition, to provide insights regarding the relationships between detected labels and ground truth labels, we categorize ground truth labels into 11 different categories.


Error Report

Result Summary detection

class name	average precision	detected number	gt number	correct detection	classification error	localization error	cls and loc error	duplication error	backdetect error	other error
🟦 bicycle	0.278	582	1039	275	51	78	60	0	91	27
🟪 bus	0.381	1087	1660	604	304	17	77	0	36	49
🟥 car	0.658	98937	102837	64054	1443	23114	1560	333	5208	3225
🟩 motorcycle	0.234	215	460	103	29	13	26	3	26	15
🟦 other person	0.000	0	1	0	0	0	0	0	0	0
🟩 other vehicle	0.000	1	85	0	1	0	0	0	0	0
🟦 pedestrian	0.379	10422	13425	5017	218	2507	489	41	1664	486
🟦 rider	0.234	257	658	150	58	10	31	2	6	0
🟧 traffic light	0.334	21319	26884	9058	256	7873	423	131	2784	794
🟨 traffic sign	0.418	28236	34724	13670	307	3961	445	22	8534	1297
🟥 trailer	0.000	0	2	0	0	0	0	0	0	0
⬜ train	0.000	0	15	0	0	0	0	0	0	0
🟧 truck	0.381	2869	4243	1602	512	97	296	3	151	208

Score Distributions

Figure 3. If users click the cell, the chart panel and image view panel update automatically with corresponding values.

Detected label types

We create 7 different detected labels to categorize every detected label based on the 6 error types from TIDE [1]. In accordance with the TIDE [1], the authors categorized the incorrectly detected labels into 6 different types: classification error, localization error, classification and localization error, duplicate detection error, background error, and missed ground truth error. However, several incorrectly detected labels do not belong to any of the categories. Therefore, we have revised the categories into 7 different types and included correctly detected labels to categorize all the detected labels. The following detected types (see fig.4) are calculated based on the two different IoU thresholds: foreground IoU threshold (tf) 0.5 and background threshold (tb) 0.1 [1].

1. Correctly detected: largest IoU and IoU >= tf for ground truth bounding box and detected class are correct.
2. Classification error: max IoU >= tf for ground truth bounding box but the detected class is incorrect.
3. Localization error: tb <= max IoU <= tf for ground truth bounding box, and the detected class is correct.
4. Classification and Localization error: tb <= max IoU <=tf for ground truth bounding box, and the detected class is incorrect.
5. Duplicate detection error: max IoU >= tf for ground truth bounding box but there is another bounding box with a higher IoU score exist.
6. Background error: max IoU <= tb for ground truth bounding box.
7. Other Incorrect errors: other detected bounding box that does not belong to the above 6 types.

From the existing categorization methods from the TIDE [1], incorrectly detected labels with the characteristic of type 7 from our category and correctly detected labels did not have their type. We categorize all the labels into one of the categories to let users easily explore all the images based on certain detected types and classes. In the post-processing step, we calculate the max IoU score for every detected label. For each detected label, we categorize it into one of the 7 detected categories based on its IoU score and its predicted class.

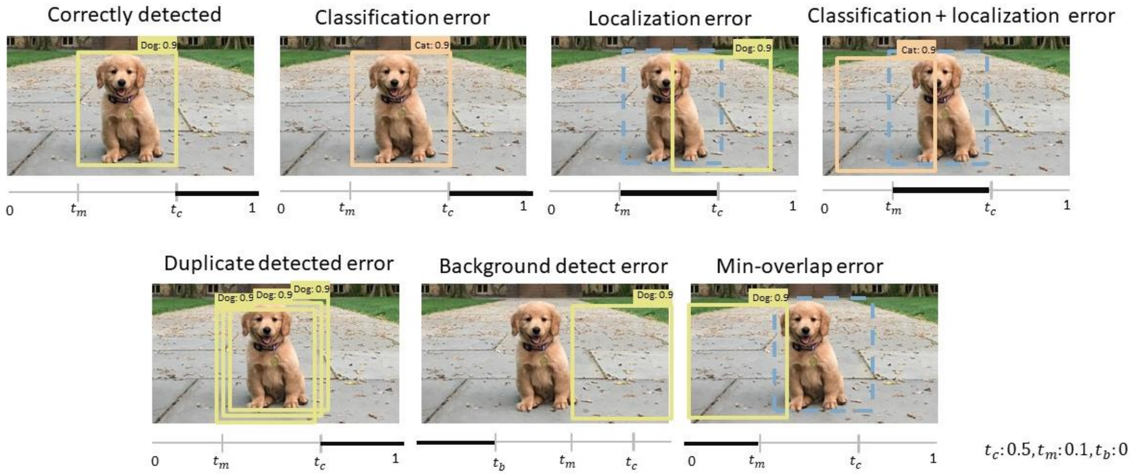


Figure 4. Detected label types

Ground-truth label types

We also create 11 different ground-truth label types (see fig.5) to categorize all the ground-truth labels based on the relationship with detected labels. When analyzing the results of the object detection model, exploring both the ground truth labels and the detected labels together may provide more diverse perspectives than when analyzing only detected labels. If both detected and ground truth bounding boxes are shown together, end-users can analyze the performance more accurately when the ground truth's relationship with detected boxes is shown than when there is just basic location information on ground truth. For example, there should be cases where a certain ground truth object is properly detected only once and another one is done several times. In that situation, by accumulating and analyzing the images of such cases, end-users can have a better understanding of the commonalities and features, which is more useful than the plainly expressed ground truth.

1. One match + Correct label: Matches with only one detected label and the matching detected label is type correct.
2. One match + classification error: Matches with only one detected label and the matching detected label is classification error.
3. One match + localization error: Matches with only one detected label and the matching detected label is localization error.
4. One match + classification, localization error: Matches with only one detected label, and the matching detected label is classification + localization error.
5. One Match + incorrect error: Matches with only one detected label and the matching detected label is another incorrect error.
6. Matches + Correct and duplicate errors: Matches with more than one label and all the detected labels are detected with the correct category
   - . All the detected labels are correct and duplicated errors, detected categories match with ground truth labels, and all IoU >= tf
7. Matches + Correct and classification errors: Matches with more than one label and all the detected labels have IoU >= tf but include correct and incorrect labels.
8. Matches + Correct and localization errors: Matches with more than one label and all the detected labels are detected with the correct category.

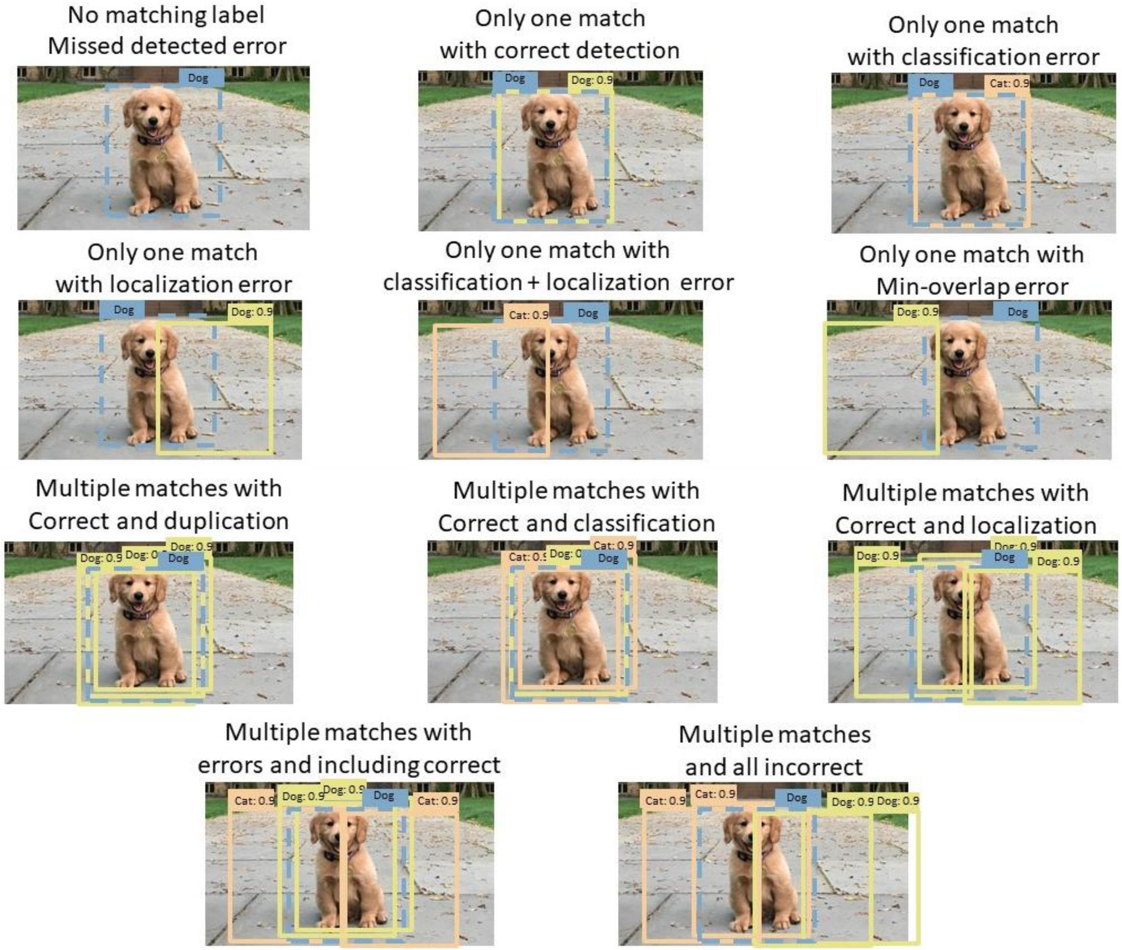
. Including detected labels with IoU >= tf and tb <= IoU < tf.

9. Matches + Include correct: Matches with more than one label and include at least one correct label.
10. Matches + Incorrect: Matches with more than one label but all the detected labels are incorrect.
11. Missed detected error: No match with the detected label.

Type 11 in the ground truth category was one of the 6 types of errors made from TIDE [1], which is an undetected ground truth not already covered by classification and localization error (false negative). TIDE [1] put this error into their 6 types of errors, but we put this false-negative labels into one of the ground truth categories because we categorize all the ground truth labels and separated them with the categories of detected labels. We use the matching index between detected labels and ground truth labels that we calculate through the post-processing steps to categorize the ground truth labels.

Based on the created categories and classes, users can click the cell of detection categories and class to see corresponding images to the cell from the detection summary table. When the users hover over each cell of the table, the letters on the selected cell turn into bold to notify the cell that the users are hovering over. The background color of the cell will also change to a darker color when the users click the cell, and the chart view and image view will update automatically based on the class and detection type that users select. [Figure 3] The cell with highlighted color will change when the users click a different cell, and the chart and image will automatically update. If the users double click the highlighted cell, the color of the cell goes back to the default color, and the chart and image view will also go back to the default view.

Next to the result summary title, there is a toggle switch button that changes between detection result and ground truth result. The ground truth summary table shows the number of each ground truth category for each class. (See fig.2) In the class column, there is a color block next to the name of each class. Each color block shows the color of the bounding box depicted on the image view panel.



No matching label Missed detected error

Only one match with correct detection

Only one match with classification error

Only one match with localization error

Only one match with classification + localization error

Only one match with Min-overlap error

Multiple matches with Correct and duplication

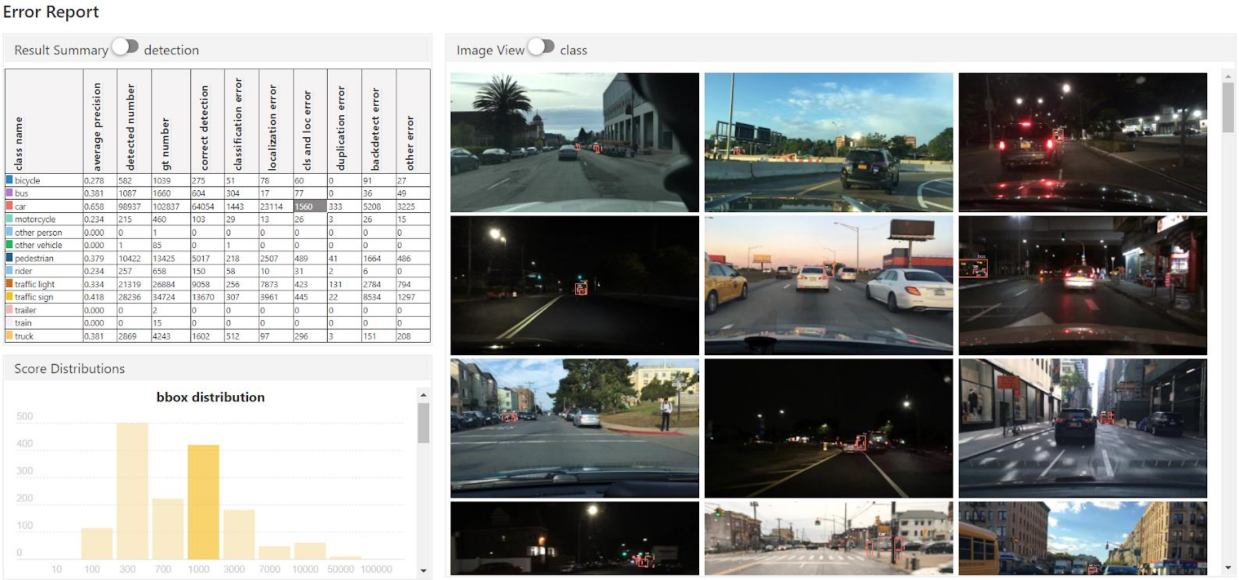
Multiple matches with Correct and classification

Multiple matches with Correct and localization

Multiple matches with errors and including correct

Multiple matches and all incorrect

Figure 5. Ground-truth label types



class name	average precision	detected number	gt number	correct detection	classification error	localization error	cls and loc error	duplication error	backdetect error	other error
bicycle	0.278	582	1039	275	51	78	60	0	91	27
bus	0.381	1087	1660	604	304	17	77	0	36	49
car	0.658	98937	102837	64054	1443	23114	1560	333	5208	3225
motorcycle	0.234	215	460	103	29	13	26	3	26	15
other person	0.000	0	1	0	0	0	0	0	0	0
other vehicle	0.000	1	85	0	1	0	0	0	0	0
pedestrian	0.379	10422	13425	5017	218	2507	489	41	1664	486
rider	0.234	257	658	150	58	10	31	2	6	0
traffic light	0.334	21319	26884	9058	256	7873	423	131	2784	794
traffic sign	0.418	28236	34724	13670	307	3961	445	22	8534	1297
trailer	0.000	0	2	0	0	0	0	0	0	0
train	0.000	0	15	0	0	0	0	0	0	0
truck	0.381	2869	4243	1602	512	97	296	3	151	208

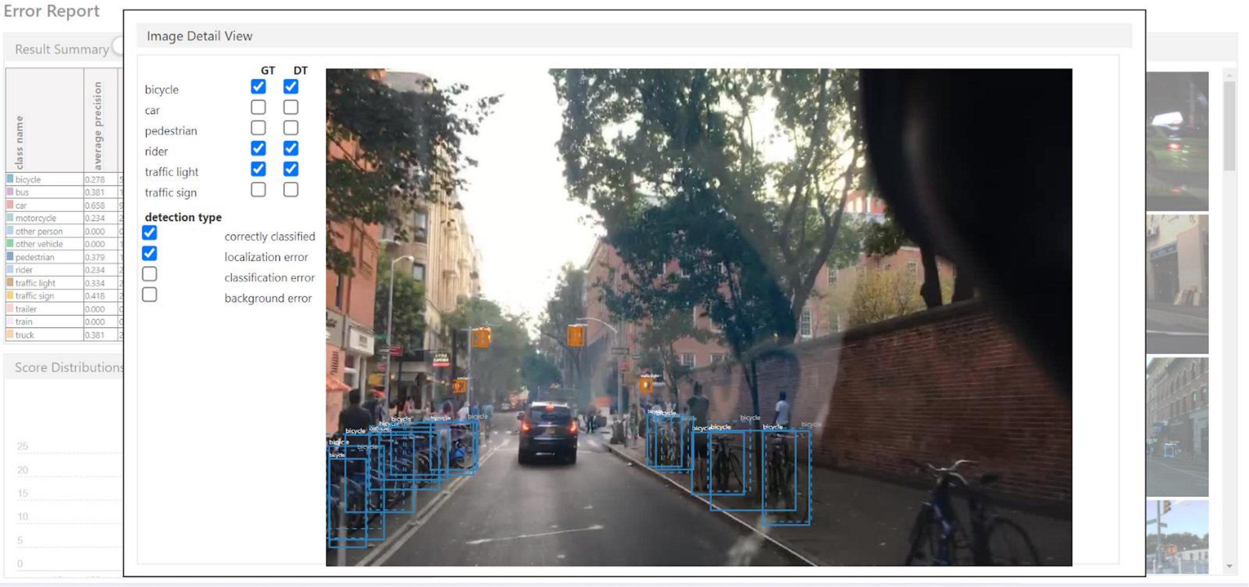
Figure 6. When users click one of the bars from the chart view panel, the image view panel updates automatically with corresponding images.

3.1.2 Chart view panel

A chart view panel is located on the bottom left of the tool (see Figure 1. B). It shows three different bar charts: the distribution of the size of bounding boxes of detection labels, the distribution of the IoU score of detection labels, and the distribution of the detection score. We select these three aspects to see correlations between different detection categories. On the default page [Figure 1], the chart panel shows three distribution bar charts for the whole detected labels, and the color of the default bar chart is light blue. The bounding box bar chart has 10 different labels, from the size 10 to 100,000, the detection score distribution bar chart has 9 different labels, from 0.2 to 1.0, and the IoU score chart has 10 different labels, from 0.0 to 0.9. The labels are created based on the minimum and maximum values of each chart value.

When the users click one of the class and detection cells from the summary table, the chart view panel gets updated based on the detection labels belonging to the selected class and detection. The color of the bar chart for the selected class and detection type is light yellow. We use different colors to differentiate between the default distribution chart and the selected class and detection chart. In the selected class and detection chart, users can select one of the bars to browse corresponding images in the image view panel, and the color of the selected bar turns darker yellow. The users can reverse their choice by double-clicking the selected bar, and the image view panel shows the whole selected and detection images, and the selected bar will go back to light yellow.

The default three distribution charts are the bounding box size chart and IoU score chart. However, there is a variation depending on the detection category. The classification error category is when the detected bounding box has a higher IoU score than the IoU threshold, but the detected category is incorrect. It is important to know how the detected category is incorrectly detected in this case. Therefore, we add one more distribution chart called the missed detected class chart which shows how the detected labels of the selected class are incorrectly classified. The labels in the missed detected class chart are the names of the classes that detection labels incorrectly classified, so the number and names of the label are different depending on the selected class. The users can also click one of the bars to see corresponding images on the image view panel. (see fig.4) In addition, the background detection error category is when the detection result detected the background. Therefore, there is no IoU distribution chart for this case because there are no matching bounding boxes between the ground truth bounding box and the detected bounding box.



Error Report

Result Summary

class name	average precision
bicycle	0.278
bus	0.381
car	0.658
motorcycle	0.234
other person	0.000
other vehicle	0.000
pedestrian	0.379
rider	0.234
traffic light	0.334
traffic sign	0.418
trailer	0.000
train	0.000
truck	0.381

Score Distributions

Image Detail View

	GT	DT
bicycle	☑	☑
car	☐	☐
pedestrian	☐	☐
rider	☑	☑
traffic light	☑	☑
traffic sign	☐	☐

detection type

☑ correctly classified
☑ localization error
☐ classification error
☐ background error

Figure 7. The modal window for each image for more detailed information.

3.1.3 Image view panel

The image view panel is located on the right side of the tool (see Figure 1. C). The default image view shows the first 500 images from the BDD 100k dataset [9]. We use 13 different colors to represent each class, and the color is shown in the summary table. To represent the detection result and ground truth result, we use a solid rectangle and dotted rectangle, so the users can see both detected bounding boxes and ground truth bounding boxes of each image at once. Also, we use a pop-up window for each image to show all the detected bounding boxes and ground truth bounding boxes on a bigger screen.

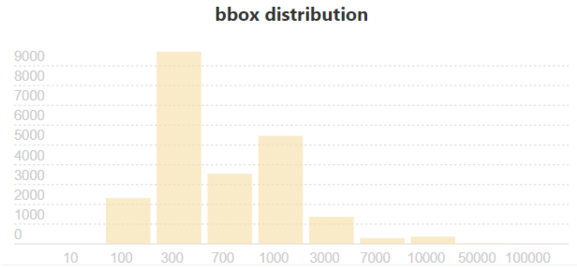
When the users select one of the cells in the detection summary table, the image view panel shows the corresponding images. For each class and detected type, we find the image index that contains the selected class and detected type. Then, we calculate the image index and the number of detected labels that belong to the selected class and detected type and sort the list of image indexes based on the number of detected labels. Therefore, in the image view panel, the order of the image is the number of detected labels that belong to the selected class and detection type. The selected image view only shows detected and ground truth bounding boxes that belong to the selected class and detection type. Users can hover over the detected bounding box to see the ground truth class, predicted class, and detection score.

At the selected image view, users can click each image for more detailed information. The modal window pops up when users click any image. As we can see in fig 5, there is a list of classes with ground truth and detected checkboxes and a list of detected error types, where users can check to see corresponding bounding boxes. When users first open a modal window, the selected class and error type from the table summary is selected as a default, and users can toggle checkboxes to see other bounding box information.

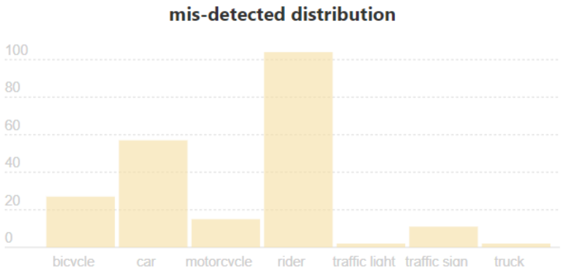
3.2 Post-processing

From the detection result of the object detection model, several post-production steps are required to have an adequate data structure for our tool. First, before categorizing the detected and ground truth labels. For each image, we compare every detected bounding box with ground truth bounding boxes to find a matching ground-truth bounding box for each detected bounding box. From every detected bounding box from one image, we calculate all the Intersection over Union (IoU) between a detected bounding box and all the ground truth bounding boxes from the same image. Then extract the max IoU value from the list of calculated IoU scores and its corresponding ground-truth label id. In this way, it is easy to check the detected and its corresponding ground-truth label. When calculating the IoU, we set the default IoU value as -1, and if the maximum IoU is -1, then it means there is no match between the ground-truth bounding box and the detected bounding box. In this case, we left the matching ground-truth id field and max IoU value field empty. After we find the max IoU value and its matching ground truth label for every detected bounding box, we use the well-known object detection post-processing algorithm non-maximum suppression (NMS) to reduce the overlapping detected bounding boxes. NMS algorithm uses the detection score and IoU value to calculate which labels to remove. It designates one label with the highest detection score from the input list as a standard and compares the IoU score with the pivot label. If the IoU score between the pivot bounding box and another bounding box is higher than the IoU threshold parameter (0.5), it deletes the labels to remove overlapping bounding boxes.

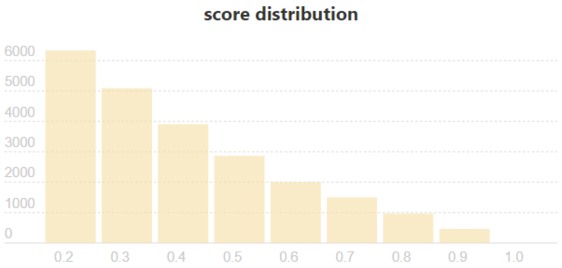
After we apply the NMS algorithm and reduce multiple overlapping labels, we calculate the mean average precision (mAP) for each class. Average precision (AP) is a widely used evaluation metric to check the accuracy of object detection models, which computes the average precision and recall curve. To calculate the precision value, we need to compute true positive values over the summation of true positives and false positives, and to calculate the recall value, we need to compute the true positive values over the summation of true positives and false negatives. In the object detection model, we consider one detection as a true positive when the predicted class is the same as the ground-truth class and also the IoU value is larger than the IoU threshold. A false positive is either when the predicted class is different from the ground-truth class or the IoU value is smaller than the IoU threshold. A false negative value will be the undetected ground-truth bounding box, and a false positive does not apply to an object detection model. We use 10 different IoU thresholds, from 0.2 to 0.7, to calculate the true positive and false positive, and calculate the mAP score for each class.



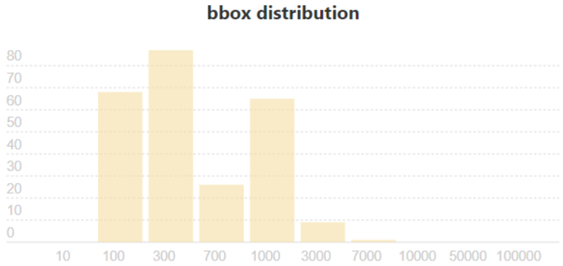
a) Car localization error bounding box distribution chart



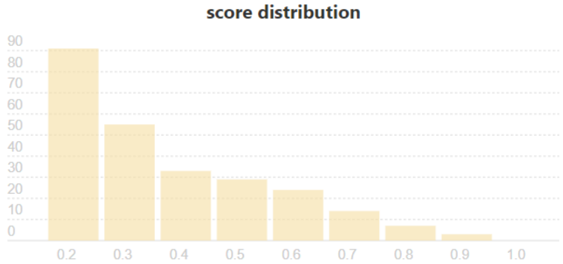
b) Pedestrian localization error class distribution chart



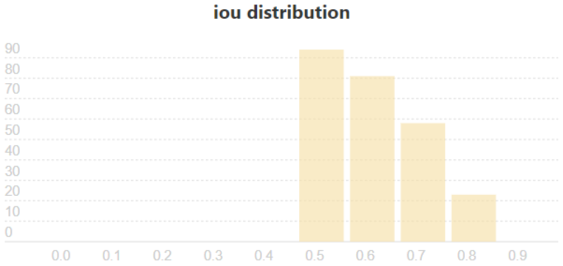
c) Car localization error detection score distribution chart



d) traffic light classification error bounding box distribution chart



e) traffic light classification error detection score distribution chart



f) traffic light classification error IoU distribution chart

Figure 8. Chart images for user scenarios

4 Use cases

In this section, we want to verify the functionality of the tool through a user scenario case. For the use case scenario, we will create a persona called Amy as a potential user of our tool. We assume that she has basic background knowledge about machine learning. She wants to use this tool to answer the questions that she has had in her autonomous driving research.

4.1 Dataset and model

For the experiment, we have chosen the BDD 100K driving dataset [9], a well-known object detection dataset, which is also known to have a very large-scale dataset with more than 10 object categories. There are a couple of experiment objectives to lead us to select this dataset: Firstly, we hope to investigate how classes with similar properties are detected/differentiated and to produce detection results from a minimum of 7 to 8 classes as well as from distinct objects. Given this dataset has 13 classes, we can perform this experiment objective without hurdles. Secondly, we want to test whether external factors influence detection results. Given this dataset has several external properties such as brightness and darkness (daylight and night-time), and occluded or blurred, it enables us to carry out the second experiment objective. We have chosen Faster-RCNN Resnet50_v1 [10, 11] pre-trained model to train and test the dataset. We fine-tuned it with the BDD 100k dataset. The total mAP value with COCO 2017 dataset was 29.3, and the total mAP fine-tuned with BDD 100K train dataset was 25.3. Classes like other person, other vehicle, trailer, and the train had only 2 to 15 ground truth labels and detected labels are usually zero. In addition, the value of the mAP calculated without the 4 classes was 36.6, which was enough for the experiment.

4.2 User scenario

We create a scenario where our persona uses our tool to answer her research questions. She will investigate the cause of errors while interpreting the results of the object detection models. Frequently occurring classification and localization errors are her main focus.

4.2.1 Finding similarities between certain errors

Classification errors.

Amy first looks for the summary view panel to view the overall performance of the Faster-RCNN [10, 11] model. She finds that there are many classification errors among the classes, so she wants to look for any common characteristics in those images or objects. First, she clicks the ‘car, classification error’ cell from figure 1. A, to see how the object ‘cars’ are incorrectly classified as other classes. In the image view panel, she checks images that have detected labels mis-detected as cars. Then from the chart view panel, she checks the classification error distribution chart, and she finds that many trucks and buses are mis-detected as cars. After looking at the images from the image view panel she first finds that many labels are incorrectly detected as cars when their ground truth labels are trucks, mostly the cases are when the objects are white vans or very small. She also finds that many of the mis-detected objects are placed in the dark or shadowed places.

Second, she clicks on the ‘bicycle, classification error’. From the chart distribution panel, she clicks bars of the large size bounding boxes from the bounding box size distribution charts to see corresponding images and finds the similarity that many objects are also placed in the night scene or shadowed. Third, she clicks the ‘traffic light, classification error’ cell to see any common characteristics among the images. Front the chart view panel, she looks at the bounding box and detection score distribution chart and finds that most of the bounding boxes have small sizes and the detection score is quite low. Also, the classification error distribution chart shows that many traffic signs are mis-detected as a traffic lights. Then she looks at the image view panel to browse corresponding images and finds that most of the objects that are incorrectly detected are commonly from night scene images and the objects are rather small or shadowed.

Localization errors

Amy then clicks the ‘car, localization error’ cell to see when and how the localization errors occur. From the chart panel view, in fig 6. a, she finds that the size of the bounding boxes is small, and detection scores are distributed in the low values (see fig 6. c). Then from the image panel view, she finds that localization errors are closely located so she clicks the image to popup the modal window to have a close look at them. After clicking ground truth and localization error together there, she can compare the sizes of the ground truth bounding box and detected bounding box in a bigger window in more detail.

After browsing several images, she finds that many localization errors frequently occurred when the objects were stuck together, and small or occluded objects tended to have more localization errors than clear and big objects. She also finds that there are several objects with higher bounding boxes that have higher detection scores, and most of the objects with localization errors are small. After that, she clicks the ‘pedestrian, localization error’ cell to see the characteristics of the corresponding images. After browsing the chart panel view, she finds that ‘pedestrian, localization error’ has a similar distribution to ‘car, localization error’ (see figure 6. C and fig 6. h) Then she looks at the image panel view and discovers that similar to the previous findings, many objects are stuck together, and the sizes are very small.

4.2.2 Finding example images for specific conditions

Mis-detection between pedestrians and vehicles and finding ground truth errors

Amy wants to explore this object detection model to answer the question of whether there are cases where pedestrians are mis-detected as vehicles due to the concerns that confusing pedestrians and vehicles are quite detrimental in autonomous driving. Then, she first clicks the “car, classification error’ cell, then looks at the classification error distribution chart, and clicks ‘pedestrian’ in the bar chart. After looking at several example images, she realizes that the pedestrian objects are placed right next to the cars and the objects are very small. She next clicks the “pedestrian, classification error” cell, and looks for classification errors as “car” (see fig 6. b), to see the objects which are detected as pedestrians, but the ground truth is cars. However, she finds out some objects from the examples were detected correctly. So, she clicks one of the images to look for more details and double-check the ground truth, detected bounding boxes, and finds that the ground truth is wrong.

Missed to detect pedestrians and vehicles

Amy also pursues answers on the question of when pedestrians or vehicles are missed to be detected due to the concerns about the safety in that situation by using the object detection model. detected properly?” To see the undetected objects, she first switches the detection summary table to the ground truth summary table by clicking the toggle button. Then, Amy selects “pedestrian, undetected”, and “car, undetected” cells to see common characteristics of the undetected objects. From the chart view panel of “pedestrian, undetected”, she finds that the size of the bounding boxes is very small and checks the corresponding images. Most of the undetected labels were on the sidewalks or the road, and there were densely placed in the images. In the case of the chart view panel from the “car, undetected” cell, she also finds that the size of the bounding boxes is also very small in this case and checks the corresponding images. She clicks one of the corresponding images to open the modal window and compares the correctly classified ‘car’ and missed detected ‘car’. In this case, lots of undetected ‘car’ objects were on the opposite side of the roads, outside of the roads, and they were very small. Therefore, based on those findings, Amy concludes that the missed detected ground truth labels are not critical to safety in driving, but she could run more training to improve the model.

5 Discussion and Future work

The objective of our research is to develop an interactive tool that helps evaluate and analyze the results of the object detection model. We believe that our tool can help non-computer science experts to evaluate the performance of the models and answer their questions by finding common characteristics in certain errors by browsing the images of their interests. Based on the analysis of the object detection results obtained from the tool, users can decide to use this model for their research or not or decide to train further to get higher accuracy. In addition, they may understand in which cases the model performs poorly. Despite our efforts herein, there are limitations in this work. This tool needs to enhance explainability and interpretability in order for users to use the object detection model with more ease and confidence. Specifically, our current work has not provided clear answers to the question of 'why' this model detected objects correctly or incorrectly. Instead, by showing the images and the results, the tool helps users to infer the black box of the model to a certain degree. Therefore, in future works, we need to improve the explainability by showing the images of users' interests such as using a grad-cam to show how a model detects an object. Future works may also include exploring chart values and images, intersecting with the ground truth and detected object categories, making it possible to browse more detailed and specific conditions.

Bibliography

[1] Boyla D., Foley S., Hays J., Hoffman J.: TIDE: A General Toolbox for Identifying Object Detection Errors. In: ECCV (2020)

[2] Hoiem, D., Chodpathumwan, Y., Dai, Q.: Diagnosing error in object detectors. In: ECCV (2012)

[3] Borji, A., Iranmanesh, S.M.: Empirical upper-bound in object detection and more. arXiv:1911.12451 (2019)

[4] Voxel 51. https://voxel51.com/.

[5] S. H. Naghavi, C. Avaznia and H. Talebi, "Integrated real-time object detection for self-driving vehicles," 2017 10th Iranian Conference on Machine Vision and Image Processing (MVIP), 2017, pp. 154-158, doi: 10.1109/IranianMVIP.2017.8342340.

[6] M. Vas and A. Dessai, "Lung cancer detection system using lung CT image processing," 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA), 2017, pp. 1-5, doi: 10.1109/ICCUBEA.2017.8463851.

[7] C. -C. Liu and J. J. -C. Ying, "DeepSafety: A Deep Learning Framework for Unsafe Behaviors Detection of Steel Activity in Construction Projects," 2020 International Computer Symposium (ICS), 2020, pp. 135-140, doi: 10.1109/ICS51289.2020.00036.

[8] M. Jain et al., "Object Detection and Gesture Control of Four-Wheel Mobile Robot," 2019 International Conference on Communication and Electronics Systems (ICCES), 2019, pp. 303-308, doi: 10.1109/ICCES45898.2019.9002323.

[9] Fisher et al.: BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning. arXiv: 1805.04687

[10] Ren S., He K., Girshick., Sun J.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv: 1506.04197

[11] TF2.0 Faster R-CNN with Resent-50 V1 Object detection model: https://tfhub.dev/tensorflow/faster_rcnn/resnet50_v1_640x640/1

[12] Google Open Images Dataset V6 https://storage.googleapis.com/openimages/web/index.html