Simon Fraser University

YOLO Traffic Light Classification Analysis

Relevant code can be found below:
https://github.com/jonathanung/traffic-yolo-analysis

Jonathan Ung (jua10@sfu.ca), Nicholas Tam (nta59@sfu.ca),
Noah Vattathichirayil (nva16@sfu.ca)
CMPT 353, Spring 2025
Final Project Report

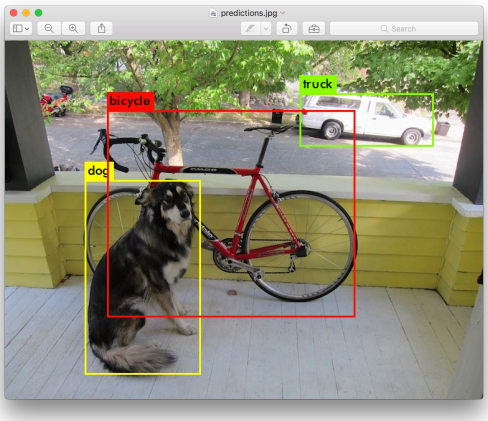
Table of Contents

Introduction

Abstract

Object detection and classification has been rapidly evolving over the past few years, as driven by Joseph Redmon's "You Only Look Once (YOLO)" CNN vision model, first introduced in 2015; it spearheaded research and development of real-time object detection, creating a world where objects could be detected with high accuracy in a reasonable timeframe, in turn making it usable in real-world applications, experiencing widespread integration into many fields such as autonomous vehicles, medical imaging, and surveillance/security. Various versions of YOLO have been introduced by various researchers over the years, with them making improvements to compute times, accuracy, and precision over each other as time goes on. Particularly, YOLOv3, v5, and v8 have each offered incremental improvements in speed, accuracy, and architectural efficiency.

As autonomous driving and advanced driver-assistance systems (ADAS) features have been propelled into the mainstream, the demand for reliable traffic light detection has grown significantly. Accurate traffic light detection enhances safety, theoretically reduces human error, and serves as a foundational component for full self-driving (FSD) equipped vehicles.



This project aims to compare YOLOv3, v5, and v8 for the task of traffic light detection, focusing on key metrics such as accuracy, precision, and inference speed. By analyzing performance across these three YOLO versions, we seek to understand how each model handles the unique challenges of traffic light detection (e.g., varying illumination, small object size, different colors), while evaluating trade-offs between model complexity and real-time performance. Through this, we should be able to gain a greater understanding of the improvements (and possible regressions) made to the YOLO model overtime.

Problem Statement

The primary objective of this project is to assess and enhance the robustness of YOLO-based object detection models for traffic light detection in real-world conditions. Despite YOLO's reputation for speed and accuracy, applications such as autonomous vehicles and smart city systems demand consistent performance under varied environmental circumstances. Factors

including fluctuating lighting (daytime versus nighttime), adverse weather (rain, snow, fog), and visual noise (occlusions, lens flare, motion blur) complicate the reliable detection and classification of traffic lights.

This project compares explicitly three iterations: YOLOv3, YOLOv5, and YOLOv8; each model will be compared to determine which version best balances accuracy, inference speed, and resource efficiency. Quantitative metrics like precision, recall, Intersection over Union (IoU), and F1-score will be used to evaluate performance, while results will be segmented based on environmental conditions to identify trends and recurring failure modes. By addressing these challenges, the project aims not only to pinpoint the limitations of current models but also to propose data-driven enhancements (such as fine-tuning and data augmentation) to improve detection reliability in safety-critical applications.

We hypothesize that each successive iteration of YOLO will yield statistically significant improvements in detection performance. Ideally, each new model should have higher mean IoUs and should have an increase in all metrics related to F1-Score.

Data Collection and Preparation

Premise

In order to collect the data expected for our analysis, we must get the formatted testing outputs from the models; thus, the following steps must be completed:

1. *[Downloading Data]* Collect the image data and truth values (initial dataset) for classified traffic lights
2. *[Initial Data Pre-Processing]* Convert the initial dataset to a format usable by the YOLO models (text label files), and split into train and test sets
3. *[Training and Inference]* Train the YOLO models to tune the weights, then test the trained model against the test data to receive its set of classified traffic lights
4. *[Labels To CSV]* Convert the labels back into a list of traffic lights detected by the model, along with confidence values, then sort the retrieved labels list by sub-dataset (sequence), then by image id
5. *[Matching CSV Data]* Match using IoU and euclidean distance to match traffic lights between testing output and the ground truth, then formatting this data to be usable in data analysis by converting to CSVs for matched lights, missing lights, and misclassified lights

The steps prove crucial to our project as the raw data only acts as an intermediate for the necessities within our data analysis, as our goal is to compare the outputs of the models in order to evaluate performance. Thus, all of these components act as essential building blocks to our data collection and preparation.

No data augmentation was performed on the base set of data. Data was verified to ensure that the model would not train on corrupted data in an attempt to avoid unintentional data poisoning.

Downloading Data

To collect preliminary data, an existing traffic light classification set must be found; LISA and BOSCH are both commonly used datasets that can be found on Kaggle. For this project, Kagglehub was used to install the LISA dataset.

Initial Data Pre-Processing

After the data was collected using Kagglehub, the annotations must be converted to YOLO format, which required processing the annotated CSV; firstly a conversion from absolute, top-left bottom-right rectangle coordinates to bounding box coordinates must be done, then all of these new labels must be converted to usable label text files for the YOLO model. Additionally, YAML files are generated to denote the paths of the datasets for training.

LISA has predefined splits for training and testing, both of which come with "ground truth" annotations. dayTrain and nightTrain were assigned to train the model, while daySequence1, daySequence2, nightSequence1, and nightSequence2, were assigned to test the model

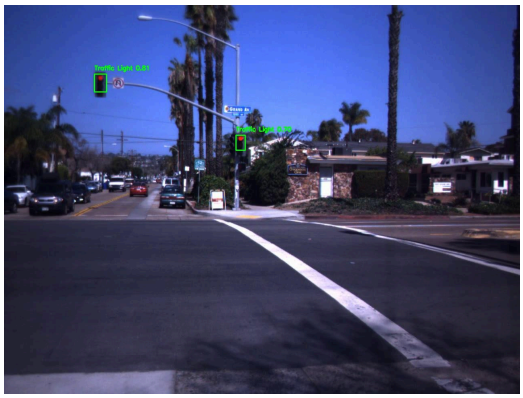
Training and Inference

$$IoU\ (Intersection\ over\ Union)\ = \frac{A \cap B}{A \cup B},\ where\ A\ and\ B\ are\ declared\ bounding\ zones$$

Training and testing operations were completed using an RTX 4080 graphics card.. By default, each YOLO model comes with a set of optimal weights and hyperparameters; the hyperparameters are implicitly defined within the repository code, while the weights are downloaded and loaded during training and testing. The sets dayTrain and nightTrain were used to train the models.dayTrain has 13 sets of clips, while nightTrain has 5 sets of clips; each set of clips has around ~8000 traffic light annotations.

To ensure data integrity given our hypothesis, boundaries and precautions were taken. All YOLO models were loaded with their set of optimal weights and had 50 epochs of training

completed to preserve similar testing conditions. Since the training compute time doesn't directly correlate to the inference time, it was not measured. All images and corresponding annotations were fed into the model during training.



Altogether, we spent around *16 hours* training all 3 models on the RTX 4080. Data was trained in the following order: daySequence1, daySequence2, nightSequence1, nightSequence2.

Inference was done afterwards, using the model weights from the trained model to perform testing. The images were parsed into the model without annotations this time. During inference, Non-Maximum Suppression (NMS) is used to filter overlapping bounding boxes and remove redundant detections. NMS ensures that the model only outputs relevant results; it is ran immediately after the model generates the raw prediction data. For NMS, the minimum IoU threshold was set to 0.45, and the minimum confidence threshold was set to 0.25.

Through using these parameters, the inference model, after running, would return all of the labels for each inferred traffic light, along with the visualizations for each image. Each traffic light label would come with a bounding box and a confidence level, and all of these outputs were compiled and aggregated to a similar format as the input. Since the inference time matters in the case of a real-world application, the time that inference took was recorded for statistical reference later using a bash script and the time command.

Labels To CSV

Labels for each dataset residing within each model needed to be recompiled into a CSV format, along with the processed LISA data to represent the ground truth. The data was simply parsed using a csvwriter to compile all the label data into a CSV for each model, along with another CSV for the ground truth.

After all the labels have been parsed into a CSV format, the data needs to be sorted by each dataset, and within each dataset, by image ID. The sorting of data ensures ease of handling, reducing the difficulty and time required to process, especially for the traffic light matching.

Matching

$$Euclidean\ Distance\text{: } d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2},\ p,\ q\ on\ cartesian\ plane$$

For each set of models, we compiled 4 CSVs:

1. Matched Traffic Lights from the model (True Positive 1)
2. Matched Traffic Lights from the dataset's ground truth (True Positive 2)
3. Misclassified Traffic Lights from the model (False Positives)
4. Missing Traffic Lights from the dataset (False Negatives)

To match each of the traffic lights to each other within an image:

1. Check IoU for matches within a threshold, then ensure those traffic lights are set as matched.
2. For any remaining traffic lights, euclidean distance is used to measure the distance between the centers of each bounding box, and the traffic lights are matched in order of closeness
3. Outliers remaining unmatched are marked as misclassified if not found in the ground truth, or as missing if not found in the model's output

This matching is necessary in order to streamline the process of the data analysis; having pre-mutated data will make it much easier to write the scripts necessary to calculate the analysis we need since more of the components used within data analysis can be computed in a matter of lines this way. Additionally, with matching comes counting; all matching is verified to ensure no traffic lights are missed. Misclassifications are also identified as they will be used to calculate error ratios further into the analysis.

We later learned it would have been a much better idea to match by Euclidean distance as a separate group of matches and verify the Euclidean matches separately. However, we managed to get around this limitation by having our scripts compare matches where IoU sits within a threshold, and for IoU being non-zero within our analysis; this point of comparison will significantly contribute later on, particularly for the analysis of YOLOv3 and YOLOv5 as those two models would show very high rates of misclassification.

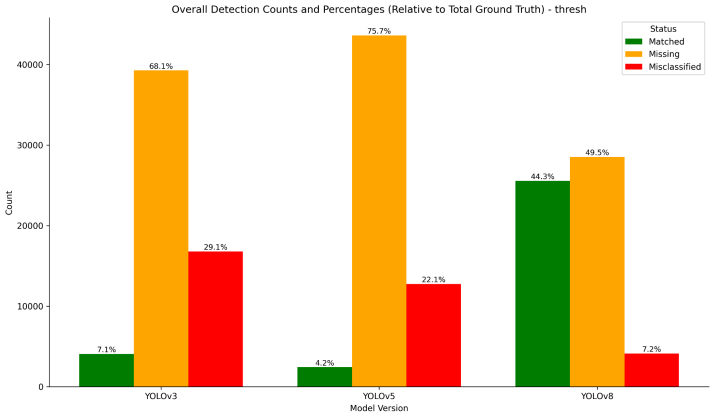
Data Analysis

For convenience, all results data has been pushed to github inside of the results/ directory. Please refer here to find the data used within the analysis.

Histograms Analysis

Overall detection counts and percentages relative to ground truth are one of the simplest yet most effective ways of understanding the models and how they have performed relative to

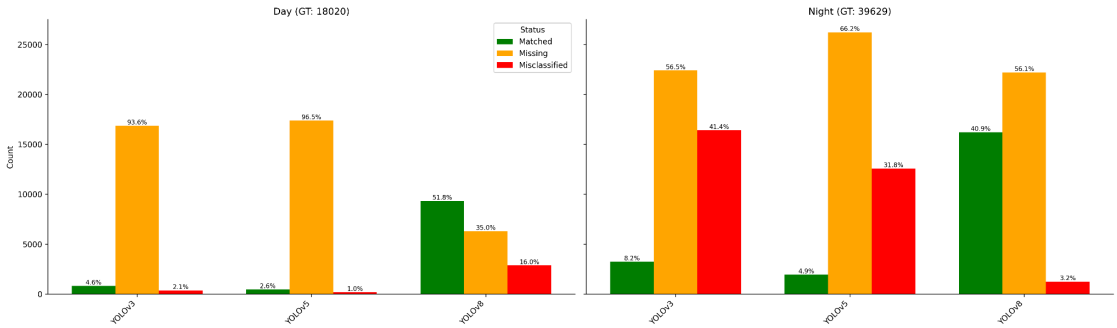
each other. Histograms showing these counts have been made for comparing both the different YOLO models, and also images from different times of day (day and night).



There were three different histograms produced with three different conditions for the YOLO model comparison, one with no IoU threshold, one with a minimum IoU of 0.45 necessary, and one with a nonzero IoU necessary (must have some IoU). Of these three histograms, the 0.45 threshold histogram is shown for simplicity purposes.

The performance comparison between YOLOv3, YOLOv5, and YOLOv8 reveals some notable distinctions. YOLOv3 and YOLOv5 displayed similar results, with histograms showing minimal differences in detection counts, suggesting they have comparable object detection capabilities. YOLOv8 demonstrated a clear edge over both predecessors, with significantly more correctly matched detections, substantially fewer missing detections, and slightly less misclassified instances. This indicates YOLOv8's superior object classification, which can be analyzed further.

Comparing the differences between time of day, it is evident that within the day, YOLOv3 and YOLOv5 miss many of the expected classifications (both looking almost the same). Meanwhile, YOLOv8 produces a healthier-looking spread, successfully classifying approximately half of the sequence. During the night, we can see that the YOLOv3 and YOLOv5 models still performed poorly but there was an increase in misclassified images. YOLOv8 was still comparatively a lot stronger than its predecessors during the night time as well. Comparing solely based on just day and night, we can notice that it is difficult to compare just off of these percentages, and that is why we will go further into recall and precision, giving more meaning to the difference of missing versus misclassified (To see more on the results of comparison, look at F1 score analysis).

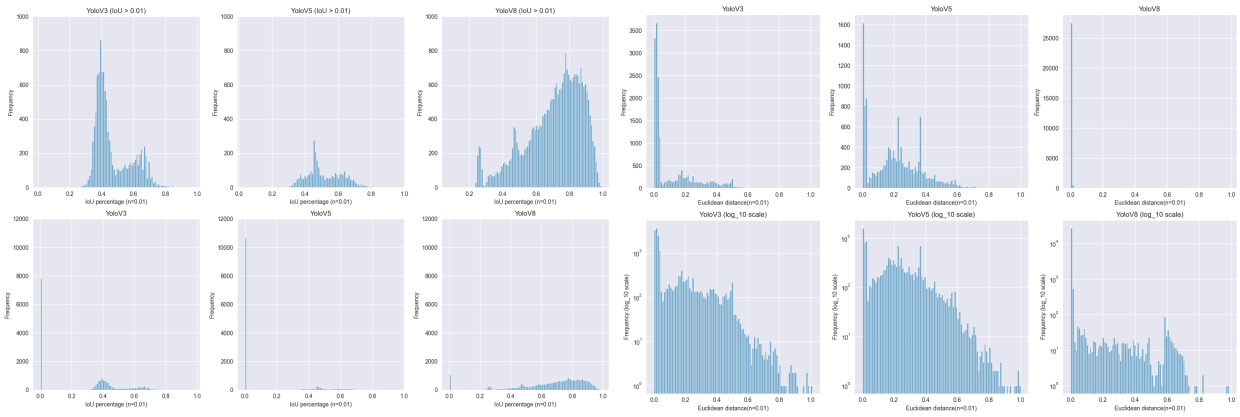


The results we obtained from this histogram analysis have motivated us to further pursue other analysis techniques justified by our judgements from what we saw here. It was clear that YOLOv8 had an edge on YOLOv3 and YOLOv5, so it was justified to do various statistical tests to prove that the difference was actually statistically significant.

Residual and Percentage Analysis

Residuals are a standard quantifier commonly used to demonstrate how much the predicted values of a model vary from reality. Data scientists commonly consider it to be a necessary metric when evaluating a model's performance. However, with YOLO predictions, this can be quite difficult as it predicts the bounding box. Though it is certainly possible to calculate residuals on each of the coordinate values of the YOLO predictions, a more informative and standard metric for the performance of object detection algorithms is needed.

IoU and Euclidean distance can serve such a purpose with the given results. Considering they measure precision, they can also serve as performance measures, imitating the applications of residuals.

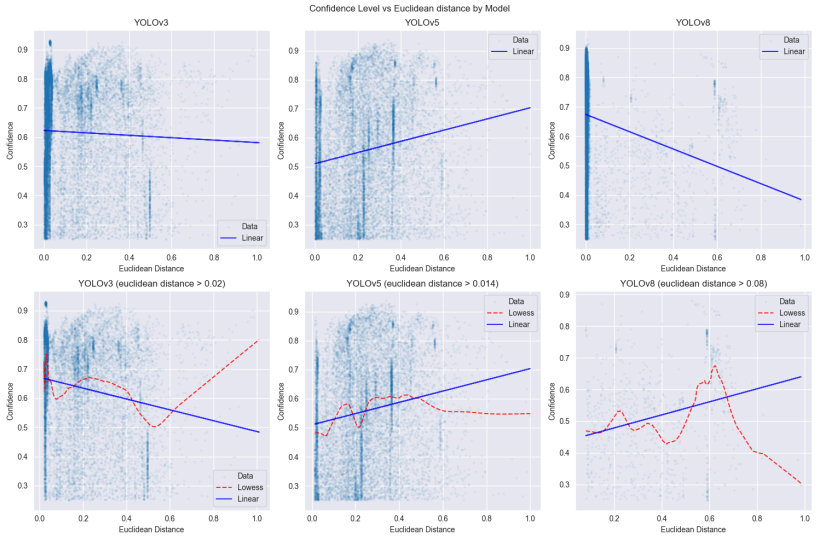


After calculating both values across all models, the data was plotted as histograms. Achieving a higher IoU frequently would imply that the model is more precise on average. Here, even through visual observation, it is noticeable that YOLOv8 experiences much greater frequency of IoU values closer to 1.0 compared to the other models; this distinction is evident within the plots which exclude the zero IoU values.

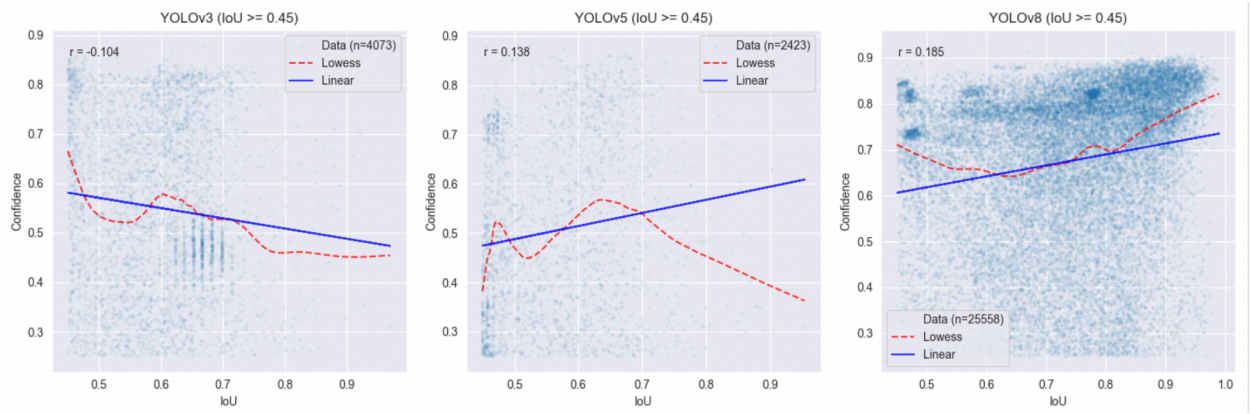
A lower Euclidean distance would imply a more precise prediction as the centers between the bounding boxes are closer. Similar to the results of IoU, it is quite obvious that v8 has a greater frequency of closer distances towards 0. However, in both thresholded and non-thresholded cases, YOLOv3 and YOLOv5 are remarkably similar in performance. If anything, YOLOv5 may have performed slightly worse.

Confidence Level Analysis

We analyzed the relationship between confidence level and Euclidean distance for YOLOv3, YOLOv5, and YOLOv8. Despite fitting linear regression lines to the data, the results indicate no strong or consistent correlation between confidence and Euclidean distance across the models.



Next, we analyzed the relationship between confidence level and IoU across the three YOLO models. Similar to the Euclidean distance results, no clear correlation was found. We produced three different versions of our plots: without any IoU threshold, with a nonzero threshold, and with a minimum IoU threshold of 0.45. For simplicity, the 0.45 IoU threshold plots for the three models are shown below.



The results of these graphs were surprising as it basically had shown us the irrelevance of the confidence level. While some linear fits may have made it seem as if there were even the slightest of correlations, upon further analysis of the points, we can see that there isn't any real correlation.

Optimal confidence thresholds for YOLOv3 and YOLOv5 remain unclear, as confidence levels appear to show little meaningful relationship with detection quality. However, for YOLOv8, a threshold around 0.65 seems reasonable, as higher confidence scores seem to actually be correlated to more accurate detections.

Analyzing the false positive and false negative trade-offs involves seeing how different confidence thresholds will affect the detection accuracy. If you have a higher confidence threshold, this leads to less false positives because only high-confidence detections are accepted, which are more likely to be correct. It also leads to more false negatives because valid detections may be thrown out with low confidence scores, especially when the model is underconfident (like in YOLOv3 and YOLOv5). If you have a lower confidence threshold, this leads to less false negatives because more detections are included, capturing more true positives. False positives will also increase because low-confidence detections might be incorrect.

Chi-Square Test

Strictly based on visual inference of the plotted data (IoU, Euclidean, confidence plots), we can already begin to draw an informal conclusion on how models can affect the precision and accuracy of predictions. However, strictly basing our data analysis on human analysis would be unprofessional and unrealistic for a data scientist. Results should be drawn from statistics and proven through tests producing statistically significant values.

```
venv(base) jonathanung@big-silver-clam-1263 Project %
python3 ./data_analysis_scripts/chi-square-test.py

Data: All included
    p-value for all: 0.0
    p-value for v3 and v5: 1.7000539179267668e-279
    p-value for v3 and v8: 0.0
    p-value for v5 and v8: 0.0


Data: Nonzero
    p-value for all: 0.0
    p-value for v3 and v5: 0.0
    p-value for v3 and v8: 0.0
    p-value for v5 and v8: 0.0


Data: Thresholded
    p-value for all: 0.0
    p-value for v3 and v5: 1.4172524541540608e-257
    p-value for v3 and v8: 0.0
    p-value for v5 and v8: 0.0
```

Within our proposal, we hypothesized that the newer the model, the better performance we should see. However, for the chi-square test, we chose a null hypothesis (H_0) that implies there is *no significant difference* between the model versions and the detection status distribution is uniform across.

Hence, the first statistical test we should implement is proving that the different models affect the ratio/proportion of matched, missing, and misclassified (MMM) even on the same dataset. This is used to statistically infer varying performance across different models. A common statistical test used for analysis of categorical data, such as MMM, is the chi-square test. This decision was made with the understanding that the data's precision and accuracy quantifiers aren't normally distributed, nor were they of equal variance.

Since our data was classified into MMM using a threshold on the IoU and Euclidean values, it was appropriate to perform separate tests of data with the threshold $IoU > 0.45$ and $IoU > 0$. For every dataset, the test was performed considering all three models at once, followed by all the possible unique pairs composed of the three models.

From the p-values, we can conclude that all are strongly affected by being in different groups (the models) as they disprove the null hypothesis with p-values of 0.0. It is worth noting that the exception to this conclusion is between v3 and v5, which are similar because v5 can be

considered an extension of v3; however, even all resulting p-values conclude a strong rejection of the null hypothesis. Hence, we can infer that the distributions of MMM, a quantifier of accuracy, are different across the group of models.

F1-Score

$accuracy = \frac{TP}{TP + FP}, recall = \frac{TP}{TP + FN}$

$F1\_Score = \frac{accuracy \cdot recall}{accuracy + recall} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$, where:

- TP = number of True Positives (Correctly Matched Lights)
- FP = number of False Positives (Misclassified Lights)
- FN = number of False Negatives (Missing Lights)

*Note that formally, it is actually called *precision* and not *accuracy*; however, in our context, using accuracy is more correct, as precision is already used to determine how close a model gets to correctly classifying a traffic light given metrics such as IoU and euclidean distance.

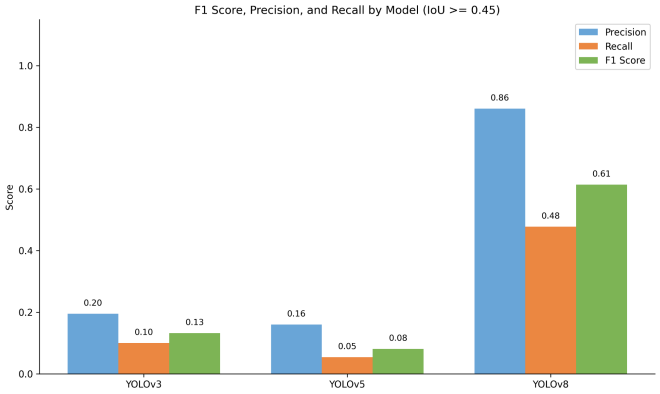
F1-Score is a measure of predictive performance, representing the harmonic mean of the precision and recall values. Using F1-Scores and the values used to calculate them, we can derive a substantial amount of information from the data.

Accuracy Analysis

The data provided allowed us to initially attempt calculating F1-Scores for three different variations of our match set: one where the F1-Score was calculated for all possible matches, one where the F1-Score was calculated for all matches where $IoU > 0$, and one where F1-Score was calculated for all matches where $IoU >= 0.45$, which was the initial threshold set during testing. However, a pattern became evident through the precision analysis, which changed our approach to analysis moving forward.

$acc_{ver} = accuracy\ for\ YOLO\{ver\}$

Initially, given all matches, we received minimally similar precision and recall scores getting $acc_{v8} = 0.98$, $acc_{v5} = 0.92$, and $acc_{v3} = 0.88$. However, given our matching thresholds which were determined during the testing



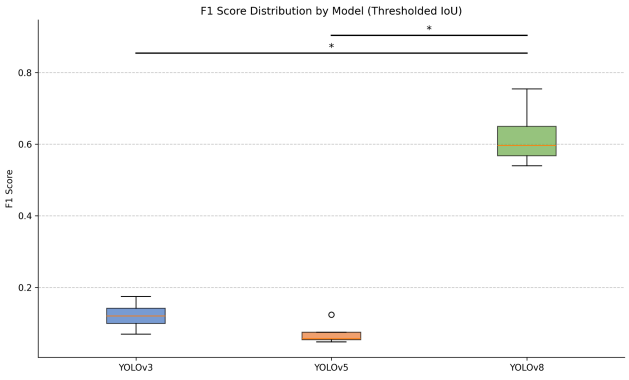
stages of the models, we quickly saw accuracies drop for YOLOv3 and YOLOv5, reaching values such as $acc_{v5} = 0.16$, and $acc_{v3} = 0.20$, while YOLOv8 remained with a relatively high accuracy at $acc_{v8} = 0.86$. This demonstrates that despite YOLOv3 and YOLOv5 seeing quite a few matches through euclidean distance, the IoU matching was inaccurate to the ground truth, meaning that the matches that occurred were referring to completely different object instances (examples found in Environmental Factor Analysis). We can confirm this occurrence through the raw counts found in the histogram analysis.

Through the accuracy analysis discovered via the histograms and the accuracy/recall scores provided when calculating F1-Scores (particularly with the performance of YOLOv3 and YOLOv5 during night time conditions) we have decided that throughout the rest of this analysis, we will target data where we have a threshold such that $IoU >= 0.45$.

F1-Score Analysis

Through our accuracy analysis, we will only be analyzing the F1 scores for the match set $IoU >= 0.45$. Notably, we can see that YOLOv3 and YOLOv5 have similar F1-Scores, while YOLOv8 has an F1-Score much higher than those in the previous YOLO models.

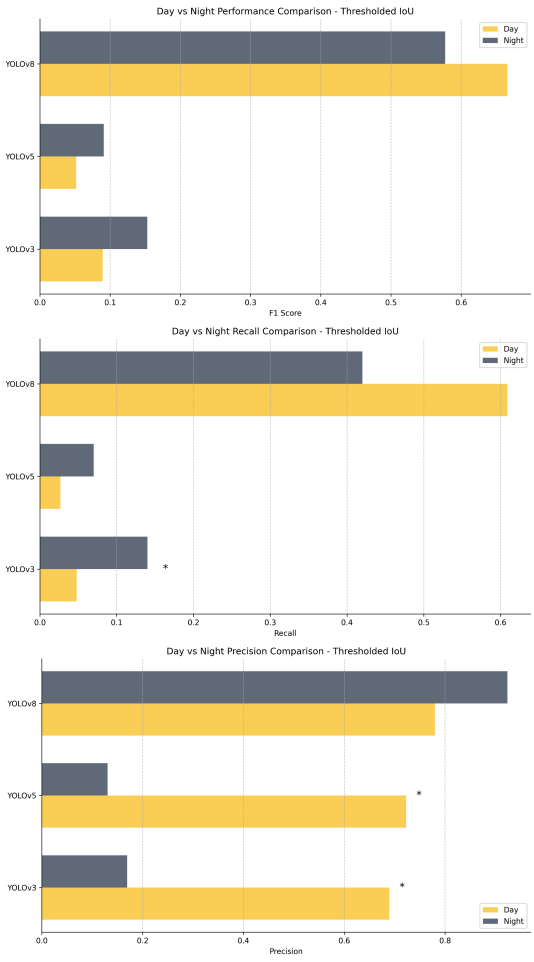
In order to simplify the t-test, our null hypothesis for the t-test is that there is *no significant difference* in the mean F1-Scores of the different YOLO model versions; essentially, we are presuming that the mean F1-Score varies marginally between each YOLO model. However, going into this t-test, we already know this will be proven wrong given the results from YOLOv8.



Through a basic t-test, we are able to see that the distribution of F1-Scores between YOLOv3 and YOLOv5 sit much lower, while no marker exists solely between those two sets of data denoting a significant difference between their F1-Score distribution. This can be largely explained by the fact that YOLOv5 is functionally a port of YOLOv3, moving from Dark-Net to Torch. Through this, we can assume no significant difference between YOLOv3 and YOLOv5's F1-Scores exist.

Through this same t-test, we can see that there is a significant difference in distribution comparing YOLOv8 and the other two models. Aside from the test results, we can easily see its boxplot sitting much higher than the other two, showing a significant improvement occurring through YOLOv8's ability to analyze traffic lights, as shown through its improved accuracy and recall scores.

Thus, for YOLOv3 and YOLOv5, we accept the null hypothesis, as no statistically significant difference was measured between the two models. However, comparing YOLOv8 to either model, we reject the null hypothesis since there is evidently a statistically significant difference measured between its and the other models' F1-Scores.



Day vs Night Performance

Another metric we can use the F1-Scores for is measuring the daytime vs nighttime performance of each model. For this t-test, our null hypothesis will be that there is no significant difference in performance between a model's performance in daytime conditions vs nighttime conditions.

Performing a t-test for each model and its daytime and nighttime F1-Scores in isolation, we can see that all of these have no asterisk in their graph, denoting that the differences in F1-Score have no statistical significance. However, given the results previously gathered from the counts histograms, we have reason to raise suspicion to these results as the misclassified count raises significantly when thresholding the IoU. Thus, we cannot easily accept this null hypothesis.

In order to measure this difference, we perform a second set of t-tests; for accuracy, our null hypothesis is that there is no significant difference in accuracy performance in a model when comparing its results in daytime vs nighttime conditions; consequently, our null hypothesis for recall is that there is no significant difference in recall performance in a model when comparing its results in daytime vs nighttime conditions.

We can see much more easily here that there is a massive difference between the accuracy achieved in daytime vs nighttime conditions, showing that the older YOLO models perform much more accurately in daytime conditions. YOLOv3 also has a significant difference in its recall performance in daytime vs nighttime, where nighttime results are proportionally much higher; YOLOv5 likely lies just in the threshold where statistical significance wasn't measured with p-values, while its difference in recall scores have enough of an effect to affect the F1-Score to no longer have statistical significance between the daytime and nighttime performance.

Overall, this shows that although F1-Scores do not differ in a statistically significant manner between any of the YOLO models, YOLOv3 and YOLOv5 achieve much better accuracy in traffic light detection, experiencing significantly more misclassifications during nighttime conditions. Additionally, since the models tend to be a lot more prone to over-classifying, their recall performance at night increases by a significant/near-significant margin. Thus, while their predictive performance measures at a similar level between these conditions, the way that their predictive performance achieves their respective scores differs significantly.

Precision Analysis

Precision for each model is defined by its ability to accurately get the correct IoU on a classification. For each of the different models, we calculated the IoU distributions for the Mann-Whitney-U test in three ways; firstly using all the IoU data points, secondly using mean IoU for each sequence, and finally using median IoU for each sequence.

Upon further examination, there is a lot of variance when calculating through all of the data points, the mean IoU for each sequence aligned with our goals the best with the Mann-Whitney-U test as we attempt to analyze the precision of each of the models; using all IoU points would be too verbose, while using median IoU points would be too robust to "outliers" when all relevant outliers have been removed through thresholding.

To compare each of the mean average values, either t-test or Mann-Whitney-U should work; we used Mann-Whitney-U since we have the means manually calculated, which t-test would do on its own anyway. For our null hypothesis, we presume that there should not be a statistically significant difference between any of the YOLO model versions when it comes to mean precision.

Our Mann-Whitney-U test discovers that there is a statistically significant difference in the means of each model's sequence when comparing to YOLOv8; just like our other tests, we see that there is not a statistically significant difference between the means present within YOLOv3 and YOLOv5. We can confirm that this conclusion is accurate; by analyzing the heatmaps of the IoU when thresholded (check Confidence Level vs IoU), we can see that the

“hot zones” that exist within YOLOv8 are pushed further down the IoU range; while YOLOv3 and YOLOv5’s “hotness” tapers off the further down the IoU spectrum you go.

Thus, for thresholded data, we can conclude that on average, YOLOv8 should achieve higher precision compared to YOLOv3 and YOLOv5. However, we cannot conclude that YOLOv3 and YOLOv5 have any statistically significant differences between each other’s precision on average.

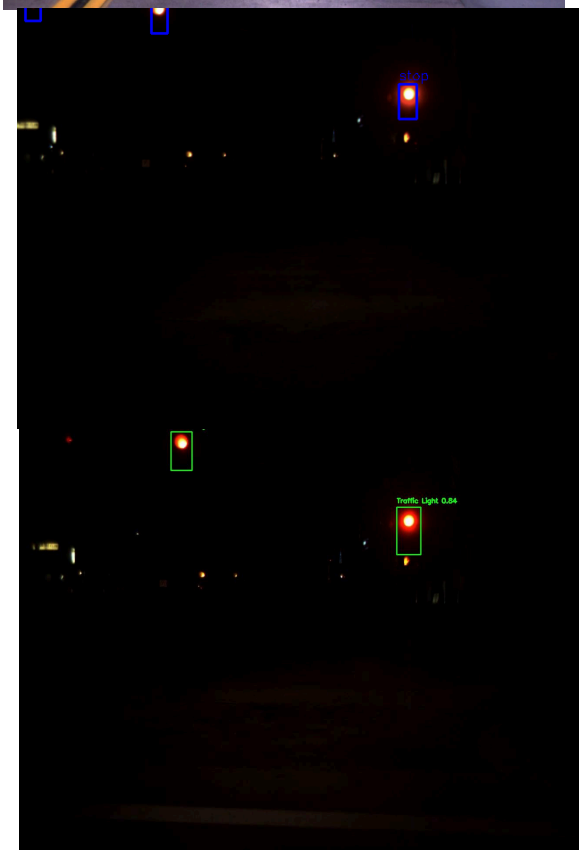
Environmental Factor Analysis

As we know from any object detection model, misclassifications can and will occur. The underlying reasons behind these errors will be analyzed by going through a variety of edge cases within the testing data’s results. By doing this, we can see what possible causes could be the driving factors in many other similar misclassifications.

When looking at the dynamic colour range of our data, daytime images generally exhibit a higher variance in color, which can come from sunlight, shadows, and diverse environmental elements. In contrast, nighttime images tend to have a broader range of luminosity, with intense light sources such as streetlights and vehicle headlights standing out against darker backgrounds. These differences in color variance and luminosity range could significantly impact the model’s ability to accurately classify objects. For daytime images, the reduction in color diversity might lead to the loss of subtle cues that distinguish traffic lights from their surroundings.Meanwhile, in nighttime images, the emphasis on luminous contrasts could result in misclassifications when bright circles are mistaken for traffic lights.

Regarding the misclassifications of bright circular things, in the YOLOv3 tested images, a road sign and the street lamp were picked up to be traffic lights. The images show what this looks like

YOLOv8 demonstrated an interesting behavior by detecting the back of traffic lights, even though it was not explicitly trained for this task. The ground truth of this image shown does not identify it as a traffic light but the model detected it as so. This is likely due to the distinct rectangular shape of the back of traffic lights, which closely resembles their front. Additionally,



the placement of traffic light, which is typically elevated and centered above the street, might have prompted the model to associate these visual and spatial cues with traffic light detection.

YOLOv8 also failed to classify a traffic light when the ground truth bounding box was only half on screen. This is understandable, as the model relies on many visual features to make accurate predictions. The left image shown is the model's prediction while the right image shown is the ground truth of the same image. With half of the traffic light out of frame, critical identifying characteristics might not have been present, making it challenging for the model to confidently detect and classify the object. This highlights the importance of full object visibility for accurate detections.

Practical Application Analysis

With the findings and conclusions in mind, it is worth considering its implications on real-world applications. Within our data, we have found statistically significant evidence that the YOLOv8 model produces both more precise and accurate predictions on average. Within the scope of our project, it would propose that object detection and Full Self Driving (FSD) vehicles should consider using newer models of YOLO, as it has demonstrated drastic improvements over its predecessors. This can help reduce the risk of misclassifications and misses, ultimately contributing to the safety of those products. However, its improved performance also translates to other use cases for YOLO, such as security, healthcare, and sports. Considering that we limited the time allocated for the models, the conclusions of our analysis reaffirm the refined efficiency of the v8 model compared to its predecessors. With real-world applications, this is an essential consideration as many of the technologies mentioned rely on efficient and accurate object detection.

Our environmental factor analysis suggests that YOLOv3 and YOLOv5 also experience a tendency to over-classify traffic lights. For instance, street lamps were often misclassified as

traffic lights. The over-classifications presented by the two models will be problematic for applications where accuracy is valued, which further affirms our hypothesis to use newer models for overall better performance.

Limitations and Future Improvements

Single dataset

During training and validation, we only had enough time to use a single dataset, LISA. This presents many limitations, including a restricted number of observations, lack of environmental variability, and potential for dataset-induced bias in the results. Considering the LISA dataset presents a narrow set of conditions (mostly clear) from real weather, the conclusions drawn may present a naive representation of the actual performance of the YOLO models. However, it is worth noting that this limitation was caused by the lack of foresight regarding the extensive resources and time consumed to train a YOLO model. This directly affected our ability to produce data as we were limited to a single device, which could train and infer the classifications. Given these limitations, future studies should experiment with other datasets to include more diversity within the training and inference; training the model on more datasets with different types of scenarios and conditions will allow the model to determine more robust weights, and inferring under varied conditions will provide more input into how the model performs in adverse edge cases. Potentially, data augmentation could be applied here to determine if a model is able to handle simulated conditions, such as rain or synthetic bright spots.

Models

As proposed, we have performed an analysis on the three commonly used models. The YOLOv3 and YOLOv5 models had similar results and performance, which was an unforeseen outcome. This is due to our lack of knowledge before the project that YOLOv5 is (essentially) a PyTorch implementation and extension of YOLOv3. In retrospect, including YOLOv11 would have presented a broader dataset to analyze and evaluate. It is also worth noting that YOLOv11 is the latest release of the YOLO models, hence producing data more relevant to modern capabilities, as YOLOv8 was released in early 2023.

We didn’t include YOLOv11 due to it not occurring to do it during our initial training and testing phase, but looking at the results, having more data to analyze from a newer model would have been very helpful in coming to more conclusions about model performance overtime. Since we ran out of time towards the middle of our project, we couldn’t fit in a train and test sequence for YOLOv11.

Implementation & Retrospect

One of the unexpected challenges that we encountered revolved around the implementation of collecting and handling the preprocessed data prior to analysis. The data could have been better handled throughout the preprocessing stage, but we didn't have the foresight to realize that many of the data groupings (such as thresholded data $IoU >= 0.45$) would be persistently relevant to our analysis. Additionally, data such as performance relative to time (using some sort of timer) was not collected; this type of analysis could have been useful in comparing the models, especially given that YOLO is a model based on classifying objects quickly. However, this was not realized early enough; by the time the models were trained and the test sets were classified, we no longer had sufficient time to spare, due to our limited hardware capable of training and opportunity to utilize it.

Conclusion

We've seen the many expected and unexpected variables that have gone into constructing the results and diagrams we've seen and analyzed. Throughout our project, from start to finish, we were able to preprocess data, train data using three separate YOLO models, sort labels and csv's, and most importantly, analyze the performance of the models. In our analysis, we used residuals, F1 scores, chi-square, and many other metrics and statistical tests to investigate how these models perform. Coming out of this study, we can see the difference in performance between the three models, YOLOv3, YOLOv5, and YOLOv8, and with confidence, we can rule YOLOv8 as the top-performing model.

YOLOv8 achieving higher mean IoU values and overall classifying the traffic lights more accurately demonstrates significant strides in improvements between the models. As real-time applications that use YOLO demonstrate progressive improvement in performance concerning classification tasks, this leads to a greater possibility that YOLO can perform adequately in safety-critical real-time applications such as ADAS and autonomous driving.

In summary, from data preprocessing to our statistical evaluation, our analysis confirms that YOLOv8 significantly outperforms its predecessors in accurately detecting and localizing traffic lights. The superior performance of YOLOv8, as evidenced by higher mean IoU values and more reliable classification accuracy, underscores its potential for integration into safety-critical real-time applications such as ADAS and autonomous driving. These findings not only highlight the steady progress in YOLO model evolution but also pave the way for further research. By exploring additional datasets and refining our methodologies, future studies can build on our conclusions to enhance object detection performance further, thereby contributing to safer and more reliable real-world implementations.

Accomplishment Statements

Jonathan Ung

- Engineered 80% of the data collection and data processing pipelines, including data unpacking, training, validation, and testing splits, and analysis pre-processing.
- Created Python analysis scripts leveraging Pandas and matplotlib to implement performance metrics tests (F1-Score and Precision) to rigorously compare the results from YOLOv3, YOLOv5, and YOLOv8 models, establishing YOLOv8 as a top performer in real-world traffic light classification.
- Wrote analysis on accuracy and precision, along with introduction and data collection and processing on the report.

Nicholas Tam

- Developed scripts for the production of analytical plots and statistical tests, including chi-square test, IoU, and Euclidean distance residuals.
- Implemented processing pipelines to convert data into required formats necessary for analysis, such as YAML, CSV, etc.
- Researched and selected YOLO performance metrics available to ensure optimal statistical tests to assess the original hypothesis.
- Authored analysis concerning residuals, chi-square, confidence relation, practical application, and methodological limitations in the report

Noah Vattathichirayil

- Pair programmed and helped with data pre-processing for transforming LISA data into YOLO readable form.
- Wrote confidence level analysis on the report, describing the relationship between Euclidean distance and IoU compared to confidence level and how that affected our further chosen analysis methods.
- Wrote histogram analysis on the report, describing the results of the test data on our three YOLO models. Mentioned the differing results in matched, misclassified, and missing data between YOLO models and between day and night.
- Wrote environmental factor analysis on the report, describing the human oriented view of how image recognition was affected through several photo examples.