ENHANCED OBJECT DETECTION:
COMPARING YOLO, DETR AND OPTICAL FLOW INTEGRATION IN AUTOMOTIVE CONTEXTS

ABSTRACT. The objective of this research is to conduct a comparative analysis of the performance of baseline YOLO (You Only Look Once) and the more complex DETR (DEtection TRansformer) models in the context of automotive object detection. A key aspect of this study is to evaluate whether the integration of optical flow as a heuristic can enhance the models' performance, both in terms of accuracy and computational efficiency.

Our research focuses on two primary contributions: firstly, to establish a comprehensive understanding of how YOLO and DETR models perform under various automotive conditions, and secondly, to assess the impact of incorporating optical flow on these models. This approach aims to leverage the motion detection capabilities of optical flow to potentially improve the real-time detection and tracking accuracy in dynamic driving environments.

To date, our experiments have involved rigorous testing of these models on diverse datasets representing a range of driving scenarios. Preliminary results indicate a notable difference in the performance of YOLO and DETR, with each model exhibiting unique strengths and limitations. While we expected the addition of optical flow to hurt accuracy but improve efficiency, it actually appears to have improved accuracy with no change to model efficiency, including in complex traffic situations with high object movement. This study is pivotal in identifying the most effective model and heuristic combination, promising to enhance the reliability and safety of autonomous driving systems.

1. INTRODUCTION

In the pursuit of refining object detection in autonomous driving systems, we confront the challenge of accurately recognizing and localizing diverse entities such as cars, pedestrians, and other vehicles within a driving context. The significance of this problem is twofold: ensuring safety in autonomous navigation and enhancing the machine's perception for better decision-making. Real-time object detection has to balance speed and accuracy, as the model must predict and recognize objects within the car's path quickly enough to act on that detection, and that detection must be sufficiently accurate. An inaccurate model in real time driving scenarios imposes significant dangers on vehicle passengers and pedestrians. Thus, we aim to explore different existing object detection models to understand how they perform within a traffic environment, weighing the key factors of efficiency and accuracy in our analysis.

For this system, the inputs are raw image frames sourced from the Kaggle Driving Video with Object Tracking dataset, which samples from the larger Berkeley DeepDrive dataset, while the outputs are categorized labels (e.g. car, pedestrian, bus, train, bicycle) and bounding box coordinates in COCO format (x, y, width, height), providing spatial localization of each detected object.

1.1. Contributions.

In the pursuit of advancing object detection within the automotive domain, our work introduces a novel approach by implementing and analyzing a hybrid model that synergizes the strengths of transformer architecture with the proven capabilities of convolutional neural networks (CNNs). We hypothesize that the self-attention mechanisms inherent in transformers can provide a more nuanced understanding of complex scenes, potentially leading to better performance than traditional YOLO models.

Contribution(s) in Application/Dataset: Our project undertakes the conversion of traditional bounding box formats to the COCO standard (x, y, width, height), facilitating compatibility with advanced neural network architectures. We have crafted a custom dataset class tailored to accommodate this format alongside appropriate labels. To ensure a balanced representation of various classes, we have employed a Multilabel Stratified Shuffle Split strategy during the dataset preparation phase.

Contribution(s) in Algorithm: Algorithmically, our experimentation spans three distinct models. We establish a baseline with the renowned YOLO architecture for its real-time detection capabilities. The DETR model, relatively unexplored in the context of autonomous driving, is examined for its complex pattern recognition proficiency. These models each balance accuracy and efficiency in opposing ways, with YOLO typically being faster yet DETR having better accuracy. Thus, the pinnacle of our experimentation is the fusion of YOLO and DETR architectures integrated with optical flow, designed to more efficiently leverage both spatial precision and temporal dynamics while integrating with the other model for improved object tracking.

Contribution(s) in Analysis: Analytically, we have adopted the bipartite matching loss coupled with a Hungarian matcher, a method not typically employed in standard YOLO-based systems. This allows for a more effective matching of predicted and ground-truth bounding boxes. The performance of our models is rigorously assessed using the mean Average Precision (mAP) metric across the test dataset, providing a comprehensive evaluation of both accuracy and reliability.

2. Background

In the realm of object detection, the evolution from classical image processing techniques to deep learning models has dramatically enhanced the capability and accuracy of these systems. The domain has seen significant milestones, beginning with R-CNN and evolving to faster and more efficient architectures like YOLO (You Only Look Once) and SSD (Single Shot MultiBox Detector). More recently, the DETR (Detection Transformer) model has introduced a novel approach by integrating transformers, traditionally used in natural language processing, into object detection tasks.

YOLO, with its real-time processing capabilities, has been extensively adopted for automotive contexts, owing to its speed and efficiency in detecting objects with a single forward pass of the network (Redmon, 2016). However, its performance in complex driving datasets often requires further improvement, particularly in terms of accuracy and dealing with small or overlapping objects.

DETR, on the other hand, has been lauded for its end-to-end object detection with transformers without the need for region proposal networks, significantly simplifying the detection pipeline (Carion, 2020). While DETR has shown promising results, its training time and complexity remain obstacles for real-time applications.

The incorporation of optical flow in object detection models presents a heuristic approach to improve temporal consistency in video streams, especially in automotive settings where movement is a critical factor. Optical flow can potentially enhance the accuracy of detecting and tracking fast-moving objects (Dosovitskiy, 2015).

DETR and YOLO provide a foundation upon which we integrate optical flow, aiming to bridge the gap between the high-speed processing of YOLO and the accuracy of DETR. Additionally, optical flow leverages the recurrent nature of video data. Where YOLO and DETR predict on a per frame basis, we use optical flow to leverage the information of past predictions in a video to inform current analyses. The overall objective is to analyze YOLO and DETR performance in an automotive context and investigate whether the inclusion of optical flow as a heuristic improves model performance in terms of accuracy and computational efficiency.

3. Related Work

DETR (Detection Transformer) represents a paradigm shift in object detection by utilizing a transformer architecture to process images holistically, contrasting with the sliding-window or region-proposal focus of conventional methods. On platforms like Kaggle, there is an abundance of implementations of DETR, such as the one by Kaggle's MRKNOWNOTHING, which offers an end-to-end detection transformer model. Our project builds upon this work, incorporating a novel modification to handle multi-label datasets—a capability not commonly explored in Kaggle contributions (source).

While DETR and YOLO are adept at image-based object detection, they do not inherently accommodate the tracking of objects over time, a critical component in automotive contexts where motion estimation is key. Our research integrates optical flow into the mix, providing the model with the capacity to estimate the motion of objects across frames, thereby enhancing prediction accuracy and enabling tracking of dynamic objects—capabilities that are essential yet missing from the standalone DETR and YOLO models. This integration of optical flow is a significant stride towards a more robust solution for real-time object detection and tracking in the realm of autonomous vehicles.

4. Approach

4.1. **Data Cleaning.** In the preparation of our object detection dataset, bounding box coordinates provided in $(x1, y1, x2, y2)$ format were converted to COCO's (x, y, w, h) format to facilitate compatibility with our model's input requirements. We faced challenges with frame extraction due to discrepancies in frame rates between the provided videos and labels. By sampling every sixth frame to match the 5Hz annotation rate from a 30Hz video, we synchronized our data with the provided bounding boxes.

Our dataset was split using Multilabel Stratified Shuffle Split, ensuring a balanced class distribution, and we employed the Iterative Stratification library, accessible at this GitHub repository. Augmentations were applied to enrich the training data, including hue and brightness variations, grayscale conversion, and flipping. Test data were kept simple with resizing and tensor conversion. A normalization step was also introduced to centralize bounding box coordinates and scale them according to image dimensions, enhancing the quality and consistency of the input data for our models.

4.2. **Loss and Performance Metrics.**

$$\mathcal{L}_{\text{Hungarian}}(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=1}^{N} \left[-\log \hat{p}_{\hat{a}(i)}(c_i) + 1_{\{c_i \neq \emptyset\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\hat{a}(i)}) \right] \tag{1}$$

To optimize the matching of predicted and ground truth objects, we adopt a bipartite matching loss as depicted in 1. This loss function allows for an optimal one-to-one assignment between predictions and ground truths, minimizing misclassifications and improving localization accuracy. The dataset comprises 100 videos, partitioned into an 80-20 split for training and testing, respectively, with frames meticulously extracted for processing.

Our choice of performance metric is the mean Average Precision (mAP), which is pivotal in object detection tasks due to its robust evaluation across varying threshold levels for classification and localization accuracy. The mAP offers a comprehensive measure, considering both precision and recall, making it an excellent gauge of a model's effectiveness in distinguishing and accurately placing bounding boxes around different object classes within the intricate environment of driving scenarios. mAP also calculates precision and recall across different thresholds and across all labels, which can then be averaged to represent the overall model performance for all objects of interest. Through this investigation, we seek to establish the comparative strengths of YOLO and DETR models, as well as the added value of bipartite matching loss, in pushing the boundaries of precision in object detection for autonomous vehicles.

4.3. **Models.**

4.3.1. *YOLO.* The most up-to-date YOLO model (YOLO v8) was used for object detection in this project. This model was initially tested on the data with its pre-trained model weights. Since this model was trained to detect and predict 80 different class labels, while the goal of this project was limited to predicting only 7 of the 80 class labels, the initial testing and validation was expected to perform slightly worse with irrelevant label predictions. The next step involved retraining the model weights for YOLO v8 using our custom training set. Each frame of the training set of videos was passed through the YOLO model in batches of 64 to undergo training and calculate optimal weights for minimization of the 3 losses calculated during training: box loss, classification loss (CLS), and distribution focal loss (DFL). Box loss focuses on the error in bounding box coordinate location (with respect to the ground truth bounding box locations), CLS focuses on the error in classifying detected objects correctly, and DFS focuses on the precision of the precision of the bounding box predictions in order to optimize a distribution of bounding box boundaries.

4.3.2. *DETR.* After pre-processing the data into the appropriate format for DETR, we loaded the pre-trained model from Facebook via the pytorch hub. As described above, we used bipartite matching loss for training and evaluation of DETR, as is done in literature. Before training on the model for fine tuning, we input our test data into the model to obtain a baseline loss for comparison. We then trained on DETR, using a maximum number of queries of 65 and a batch size for the dataset of 8. This was chosen due to a maximum of 45 bounding boxes within the training data, plus room for some more boxes to be detected for test. Our training fine tunes the pre-trained DETR model to work specifically for our traffic video dataset, and we aimed to observed a reduction in loss. After training the DETR model to minimize the loss function described above, we use the fine tuned model for test-set object detection, as well as to be incorporated into the optical flow heuristic.

4.3.3. *Optical Flow Integration.* Optical flow is computed using the `opencv` library in Python. We computed the trajectory of a bounding box from one image to the next using the four corner points of the box. This integrates the sequential nature of video data into predictions.

We designed a heuristic for full video data to integrate with the chosen models. This differs from standalone DETR or YOLO, which predict on a frame of a video. The general goal of this is to use the chosen model as little as possible. We want to reduce the overall computational cost by limiting the need to execute the chosen model (YOLO or DETR), so optical flow can be used to track bounding box movement instead. After a certain number of frames of just optical flow, we then check the chosen model to see if the optical flow box predictions still align. As part of our heuristic, we define a `padding` variable, which begins at the value of 1. For the first frame of the video, we use the chosen model to predict bounding boxes. We then predict for subsequent frames using just optical flow, but check back with the chosen model after `padding` number of frames. When padding is 1, this implies we use the chosen model at every frame. However, an additional aspect of the heuristic is our approach to this variable. Every time we check optical flow against the chosen model, we compare the bounding boxes with box IOU (intersection over union). If the box IOU is above a threshold, we imply that the optical flow bounding boxes are sufficiently close to the model. At this step, we increase the padding variable by 1 (i.e. padding could be increased from 1 to 2 to 3 and so on, increased the number of steps before we check the chosen model. However, if the box IOU is below the threshold, we reset the padding back to 1.

5. Experimental Results

5.1. **Data Statistics.** We worked with the "Driving Video with Object Tracking" dataset, which is a subset of the Berkeley DeepDrive dataset. This dataset came with 1000 videos and label csv file, which includes one row for every bounding box in every frame (sampled at 5hz from 30hz) for each video. From this, we randomly sampled 100 videos, 80 for training and 20 for testing. Each set had an even distribution of object labels. The labels included: car, pedestrian, truck, bus, bicycle, rider, other vehicle, motorcycle, other person, trailer, and train. However, due to ambiguity in some labels, we consolidated our labels into: car, person, truck, bus, bicycle, motorcycle, and train. Trailer and other vehicle were dropped due to poor labeling and infrequency, such as a grocery shopping cart being labeled "other vehicle."

	Pre-Trained	Fine Tuned
MAP	0.122	0.17

Table 1. Comparing YOLO Results using MAP

5.2. **Finetuning YOLO.** As mentioned, YOLO was initially run on the test set of videos before finetuning the model weights to our specific use case of video data. This initial performance, as expected, was slightly worse than what was obtained after training. As seen in Table 1, this initial mean average precision (mAP) improved from 0.122 to 0.17.

	Pre-Trained	Fine Tuned
Loss	3.45	0.88

Table 2. Comparing DETR Results using Bipartite Matching Loss

5.3. **Finetuning DETR.** The DETR model without finetuning performed very poorly on the dataset and was unable to identify bounding boxes for object detection. After finetuning, we saw a notable improvement in performance, with the loss decreasing from 3.45 to 0.88 as seen in Table 2.

	YOLO	YOLO+OF	DETR	DETR+OF
MAP (Full Test set)	0.17	0.290	0.39	0.407
MAP (One Video)	0.369	0.366	0.410	0.428
Time (s) (One Video)	2.69	3.07	6.20	6.57

Table 3. Cross Model MAP Results (Fine Tuned Models)

5.4. **Optical Flow Integration.** After training the models, integrating optical flow into the overall architecture resulted in improved performance overall. Although it improved for both models, the improvement was more drastic for YOLO, improving the mAP from 0.17 to 0.2896. DETR improved by far less (0.39 to 0.407). We see this in the first row of Table 3, for full test set results.

We also chose to explore model time complexity by exploring each of the models for one video in the data set. This was done to mimic a real-time model, which is just one "video" with object detection. We used the same video, which had 101 frames (sub-sampled from a 20 second video), across each models, finding the MAP and timing for each model. As expected, YOLO performed better in terms of speed, both with and without optical flow integration. We can see the single video results in Table 3

Finally, we can see a clear understanding of the success of our DETR and YOLO models on object detection as they occur in sampled images. In Figure 3, we see that both DETR and YOLO predictions actually match up well with the true bounding boxes. This occurs for images at night, with objects at a distance, and many objects to detect.

Overall, there are some notable conclusions from our results. First, we see an imrpovement for both YOLO and DETR from pre-training to finetuning. This means that our trained models are better suited for the traffic dataset we are working with. The images show this as well, as much of object detection relies on the qualitiative aspect of actually seeing the object detection succeed in the images. This gives us an understanding of how these models may accurately detect images in a real environment. While these are notable results on their own, the incorporation of optical flow provides an enhances analysis. The optical flow model does not improve efficiency as expected. However, it has either

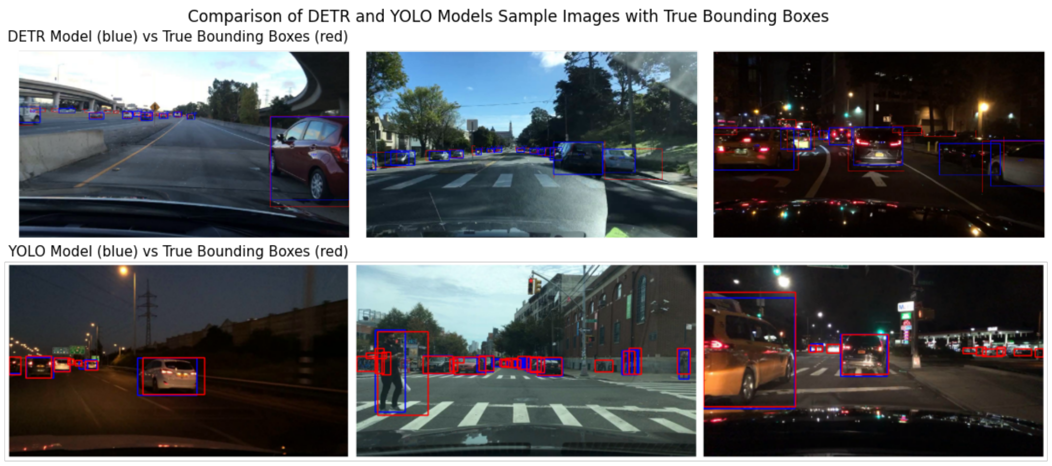
Comparison of DETR and YOLO Models Sample Images with True Bounding Boxes
DETR Model (blue) vs True Bounding Boxes (red)
YOLO Model (blue) vs True Bounding Boxes (red)

FIGURE 1. Sampled Images with bounding boxes

very similar or improved results on MAP compared to respective standalone models. The time similarity is likely due to a weakness in the heuristic, where the chosen model is being used at most iterations. We aimed to reduce its use to improve efficiency, but this does not happen. We can continue to explore the optical flow heuristic and change the thresholds as well as the mechanism for determining padding values, to hopefully achieve the results we expected. As an additional note, we also saw a remarkable improvement in the MAP value for YOLO with optical flow. We compute MAP for DETR and any optical flow models with a custom built MAP function. However, YOLO has MAP built in. This could explain the difference in metrics as well. These factors are explored in the discusion section.

Overall, our results, both qualitative and quantitive, show successful object detection for each model, with DETR consistently performing better but slower, as was expected. The optical flow integration works but does not impact results significantly.

6. DISCUSSION

Our model had a few key limitations, including training time and model comparability. Because the YOLO and DETR models were pre-trained, and we used packages for each to conduct fine-tuning, we used each models commonly used loss function. YOLO had MAP built into it, but DETR did not. Additionally, their loss functions differ slightly. In the future, we aim to develop more thorough metrics for an enhanced comparability of the data.

Additionally, we hope to train on a larger set of data. In this case, given the limited time and complexity of the problem at hand as well as limited resources, the data set used was only of 100 driving car videos, 80 for training and 20 for testing. If presented with more time for more training, the mAP scores for both models could be expected to improve significantly, as more training data could be expected to reduce the bias of the model's performance.

Beyond improving metrics and training, we also would hope to expand upon the optical flow heuristic. Currently, optical flow does not improve time, which we believe is due to frequent use of the chosen model. To get the true benefits of an optical flow addition to the model, we need to spend significant time designing the heuristic so that optical flow can be used more and the chosen model will not be computed as much. We hope to maintain the strong accuracy the optical flow model provides, while seeing the improvmeent in efficiency we expected.

Finally, we can extend this problem in the future to further mimic a real-time environment. With a better dataset and further exploration of the sequential video data, we can see how these models truly perform in a more realistic setting. This differs from our primarily frame-based approach to object detection. Integrating these models into real-time data will give us the best sense of its impact on object detection and the improvement of it for autonomous driving.

References

Carion et al., ”End-to-End Object Detection with Transformers,” 2020
Dosovitskiy et al., ”Flownet: Learning Optical Flow with Convolutional Networks,” 2015
Kuklin, Maxim, ”Optical Flow in OpenCV (C++/Python),” January 2021.
MRKNOWNOTHING. (2020, July 5). “End to end object detection with transformers: DETR”. Kaggle. [Link]
Redmon et al., ”You Only Look Once: Unified, Real-Time Object Detection,” 2016
Skalski, P. (2023, December 7). Track and Count Objects Using Yolov8. Roboflow Blog. [Link]
Ultralytics. (n.d.). Track. Track - Ultralytics YOLOv8 Docs. [Link]

APPENDIX

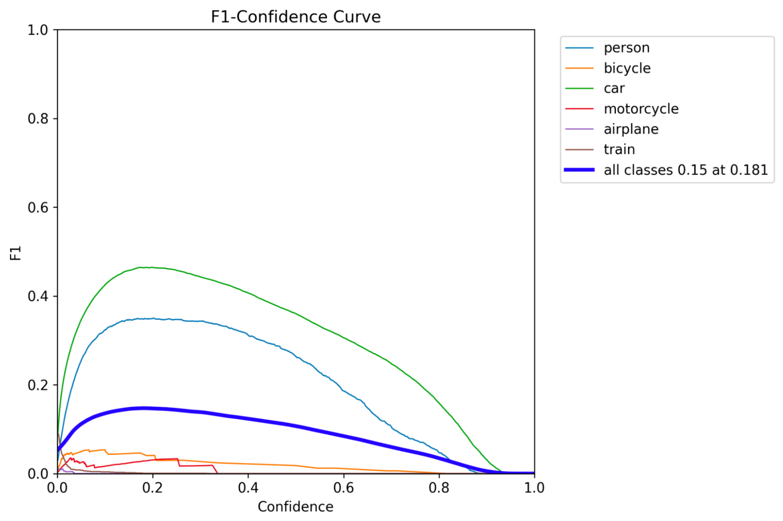

FIGURE 2. F-1 Confidence Curve of YOLO

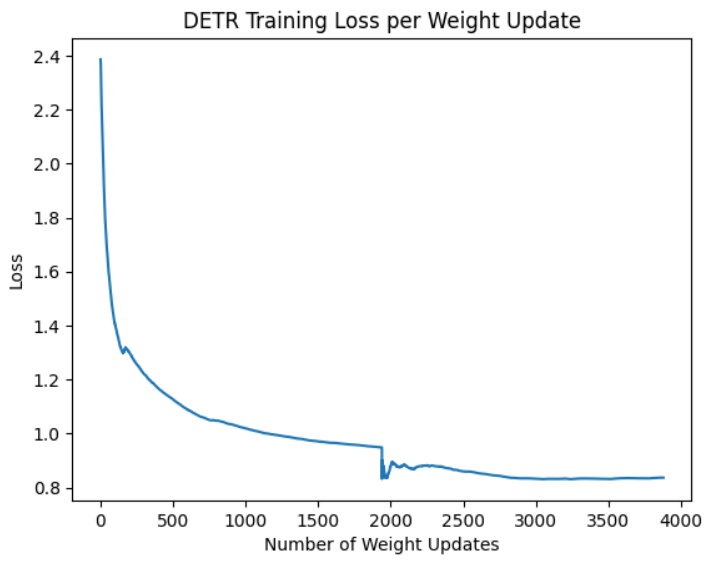

FIGURE 3. DETR Training Loss per Weight Update