ENGR E533: DEEP LEARNING SYSTEMS

FINAL PROJECT

EXPLAINABLE AI FOR OBJECT DETECTION IN AUTONOMOUS DRIVING

RITIKA SHRIVASTAVA

RISHIKESH KAKDE

SARANSH SINGH

INDEX

1. INTRODUCTION

The autonomous driving revolution stands at a critical juncture, where technological innovation meets real-world safety and reliability. At the core of this transformation lies Object Detection – a fundamental capability that determines a self-driving vehicle's ability to navigate complex and dynamic environments. Despite significant technological advances, a persistent challenge remains: the black-box decision-making processes of artificial intelligence systems that power these vehicles.

Our project addresses this gap by conducting a comprehensive comparative analysis of object detection algorithms through the lens of explainable artificial intelligence (XAI). By leveraging the Udacity Self-Driving Car Dataset, we aim to not only measure the performance of cutting-edge neural network architectures but also highlight the reasoning behind their detection capabilities and subsequently their decision making in autonomous driving.

The study will systematically evaluate four prominent object detection models: YOLOv8, Faster R-CNN, Detection Transformer, and RetinaNet. Each of these represents a unique approach to the complex challenge of real-time object identification and localization. Through meticulous transfer learning and training on a diverse dataset capturing the scenarios of road environments, we will assess their performance using industry-standard metrics like mean Average Precision (mAP), Intersection over Union (IoU) and Intersection over Foreground (IoF).

What distinguishes this research is our deep dive into the interpretability of these models. By employing advanced XAI techniques – including Grad-CAM and Saliency Maps, we will generate insights into how these neural networks perceive and prioritize visual information. This approach goes beyond traditional performance metrics, offering a transparent view into the cognitive processes of artificial perception.

Link to our repository: https://github.iu.edu/ritshriv/dls-final-project/tree/main

2.1 PROBLEM

The rapid advancement of artificial intelligence in autonomous driving technologies is accompanied by a fundamental challenge: the lack of interpretability in complex neural network models. Current object detection systems, while increasingly accurate, operate as opaque "black boxes" that provide little insight into their decision-making processes.

The primary challenge addressed by this research is the limited understanding of how deep learning models for object detection arrive at their conclusions. Despite the impressive performance of neural networks in identifying objects critical to autonomous driving - such as vehicles, pedestrians, and traffic signs - there remains a significant gap in:

- **Model Transparency** Understanding the specific features and reasoning that drive object detection decisions.
- **Interpretability** Providing clear, human-comprehensible explanations for why and how objects are identified in complex driving scenarios.

2.2 MOTIVATION

Our study is motivated by the need to:

- Compare different neural network architectures through the lens of explainability
- Visualize and analyze the internal decision-making mechanisms of deep learning models
- Enhance understanding of AI decision-making in autonomous driving object detection

The study holds significant implications for:

- Improving trust in autonomous driving technologies
- Enhancing the debugging and refinement of neural network models
- Supporting the development of more interpretable AI systems
- Providing researchers and engineers with deeper insights into machine learning decision-making

3. LITERATURE REVIEW

Object detection is one of the most pivotal tasks in computer vision, with applications ranging from autonomous driving to video surveillance. In this field, Faster R-CNN has become a cornerstone for real-time object detection. Proposed by Shaoqing Ren et al. (2015), Faster R-CNN introduces the concept of Region Proposal Networks (RPNs), which significantly enhance the efficiency of generating region proposals by using convolutional networks rather than the traditional selective search. This method allows for end-to-end training, which is a major advancement over previous approaches. Faster R-CNN revolutionized the way real-time object detection could be done by integrating both object localization and classification into a single network architecture. This model set the stage for modern object detection algorithms, enabling them to operate in real-time while maintaining high accuracy and precision. The innovation of RPNs also helped overcome the computational bottlenecks associated with earlier methods, which relied on manually crafted features.

Another significant development in the area of deep learning for image analysis is the Grad-CAM (Gradient-weighted Class Activation Mapping) technique introduced by Ramprasaath R. Selvaraju et al. (2017). Grad-CAM offers a way to generate visual explanations from deep convolutional networks by using gradient-based localization techniques. The method allows users to understand which parts of an image are being used by the model to make predictions, which is critical for improving model transparency and interpretability. This is especially important in high-stakes applications like medical imaging or autonomous driving, where understanding the model's decision-making process is essential for ensuring safety and reliability. Grad-CAM has enabled a broader understanding of how convolutional neural networks (CNNs) operate internally, particularly in complex tasks such as image classification and object detection.

Simonyan et al. (2013) explored the visualization of CNNs, contributing further to the interpretability of deep learning models with their work on saliency maps. By using techniques to visualize the most relevant regions of input images for classification tasks, they showed how CNNs learn to focus on certain features of an image, providing a clearer picture of how networks arrive at their decisions. This line of work helps demystify the inner workings of CNNs, which are often considered "black boxes." Saliency maps, along with

other visualization techniques, help researchers and practitioners understand which parts of an image influence classification outcomes, improving model trustworthiness and fostering a better user experience. This is particularly critical in domains where model errors can have significant consequences, such as healthcare and autonomous systems.

As the application of deep learning extends to more complex fields like autonomous driving, the demand for explainability and transparency in AI models grows. Marchisio et al. (2021) highlight the importance of explainable deep neural networks (XAI) in the context of autonomous driving systems. They argue that explainability is crucial for the adoption of AI technologies in autonomous vehicles, as human operators need to trust the system's decisions, especially in critical scenarios. The authors emphasize the development of transparent and interpretable models that can provide real-time insights into their decision-making processes, enhancing both safety and public trust. Explainable AI is particularly pertinent in autonomous driving, where decision-making needs to be understood not only by developers but also by regulatory bodies and end-users, who may need to intervene in certain situations.

While advancements such as Faster R-CNN have enabled real-time detection with high precision, methods like Grad-CAM and saliency maps provide essential tools for understanding how these models operate. In applications like autonomous driving, where safety and trust are paramount, the need for transparency has never been more urgent. As research continues to evolve, the integration of explainability into AI models will be essential for their broader acceptance and deployment, ensuring that they are not only effective but also understandable and reliable.

4. APPROACH

We trained 4 Object Detection Models on the Udacity Self Driving Car Dataset containing 15,000 images and 12 classes (obstacles, biker, car, pedestrian, trafficLight, trafficLight-Green, trafficLight-GreenLeft, trafficLight-Red, trafficLight-RedLeft, trafficLight-Yellow, trafficLight-YellowLeft, truck). The dataset is in the COCO (Common Objects in Context) format (a JSON structure that governs how labels and metadata are formatted for a dataset) (https://public.roboflow.com/object-detection/self-driving-car/3). Our focus was on the resized versions (512 x 512) of the images to reduce computation overhead and speed up training time.

In terms of preprocessing, we only applied normalization to the images and ensured all image tensors have the same shape.

Our aim was to study how different models perceive images and make predictions based on what features they find to be significant for detecting an object which in turn would help an autonomous vehicle make decisions.

We evaluated our models based on Mean Average Precision and Intersection over Union. To further understand how the objects are detected, our team leveraged Grad-CAM and Saliency Maps to explain the model's predictions along with generating intermediate representations of the feature maps.

5. METHODOLOGY

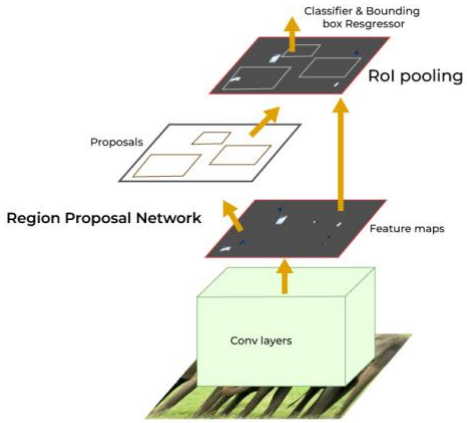
5.1 MODELS

5.1.1 Faster R CNN

Faster R-CNN: A State-of-the-Art Object Detection Architecture, introduced in 2015 [1], is an advanced object detection architecture that significantly improves upon its predecessors in the R-CNN family. This unified architecture not only detects objects within images but also precisely locates them, combining the power of deep learning, convolutional neural networks (CNNs), and region proposal networks (RPNs).

Faster R-CNN consists of two main components:

1. Region Proposal Network (RPN)
2. Fast R-CNN detector



Both components share a common CNN backbone, which serves as the foundation for feature extraction.

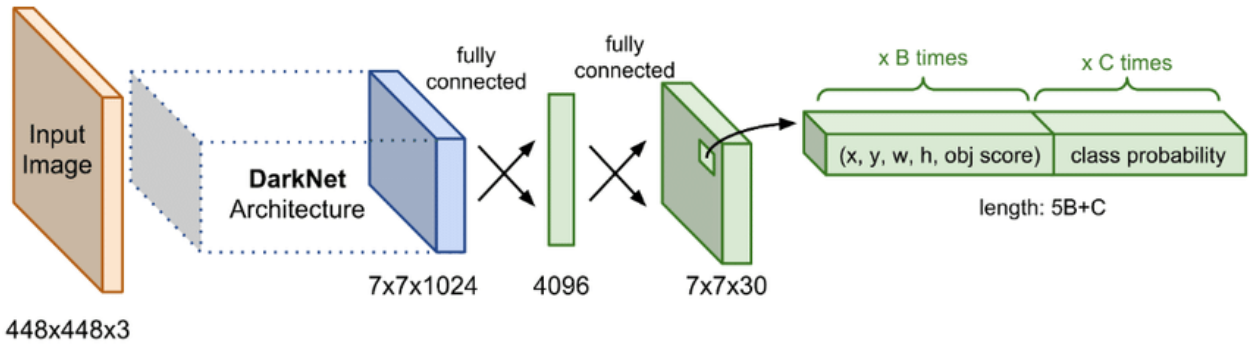
CNN Backbone: The CNN backbone is the initial layer of the Faster R-CNN architecture. It processes the input image to extract hierarchical feature maps, capturing both low-level features (e.g., edges and textures) and high-level semantic features (e.g., object parts and shapes). The shared computation significantly reduces processing time and memory usage.

Region Proposal Network: The RPN is a key innovation in Faster R-CNN, replacing the traditional CPU-based Selective Search algorithm used in earlier R-CNN models. It generates potential regions of interest (ROIs) using a convolutional-based network, reducing proposal time from 2 seconds to just 10 milliseconds per image.

Fast R-CNN Detector: The Fast R-CNN detector processes the region proposals suggested by the RPN.

Training Process: It involves fine-tuning a pretrained Faster R-CNN model (fasterrcnn_resnet50_fpn) with a custom number of classes. The roi_heads.box_predictor is replaced to adapt to the specific task, and an Adam optimizer with a StepLR learning rate scheduler is used to update model parameters. Training spans 12 epochs, where images and targets are passed through the model to compute the loss, and gradients are updated via backpropagation. A custom LossAverager tracks both training and validation losses for each epoch. Validation is performed by evaluating the model on a separate dataset, computing the average loss without updating weights. While effective, the training script could be improved by fixing the misuse of the scheduler, ensuring robust data handling, and incorporating evaluation metrics like mean Average Precision (mAP) for better performance tracking.

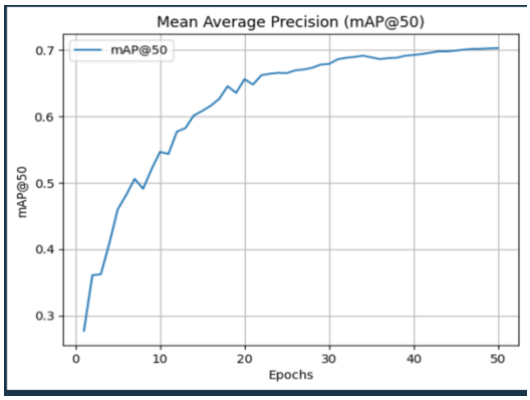
5.1.2 YOLOv8



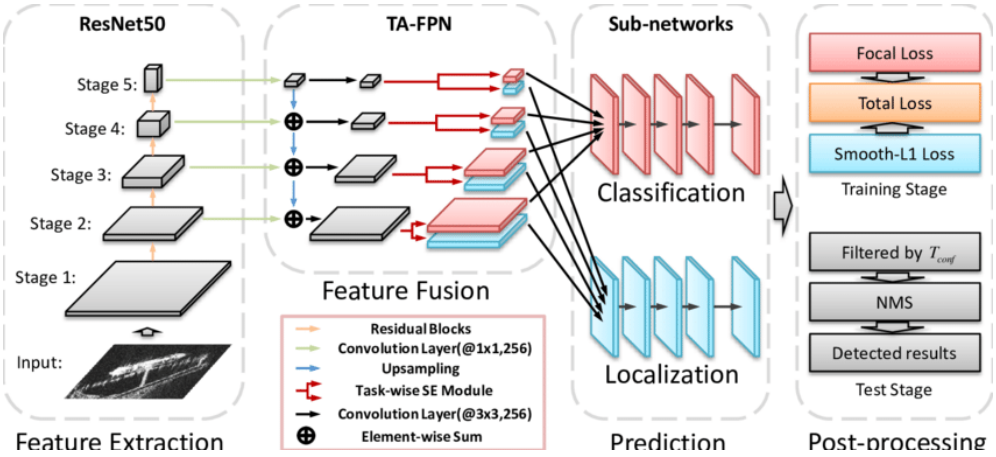
YOLO (You Only Look Once) is a highly efficient and real-time object detection framework that processes object localization and classification in a single forward pass. Unlike two-stage detectors like Faster R-CNN, YOLO simplifies detection by treating it as a regression problem, enabling remarkable speed without compromising accuracy. In this project, we leveraged YOLOv8, the latest iteration, which features an anchor-free design and dynamic modular architecture. These advancements improve prediction accuracy and reduce computational overhead. The pre-trained YOLOv8n model was adapted to our custom dataset, which included traffic elements such as pedestrians, vehicles, and traffic lights, annotated in COCO format. The model's inherent ability to predict bounding boxes and class probabilities directly from feature maps allows for seamless integration into real-time applications like self-driving cars.

The training process involved fine-tuning the YOLOv8n model using a combination of bounding box regression loss, objectness loss, and classification loss. Data augmentation techniques such as flipping and scaling were applied to make the model more robust to variations in the environment. The learning process was optimized with an adaptive learning rate scheduler for faster convergence. Validation was performed using metrics such as mean Average Precision (mAP) at various Intersection over Union (IoU) thresholds to evaluate the model's detection accuracy. This streamlined training approach and YOLO's inherent real-time detection capability make it a strong choice for dynamic, latency-sensitive tasks like traffic analysis in autonomous systems.

After Training:-



5.1.3 RetinaNet

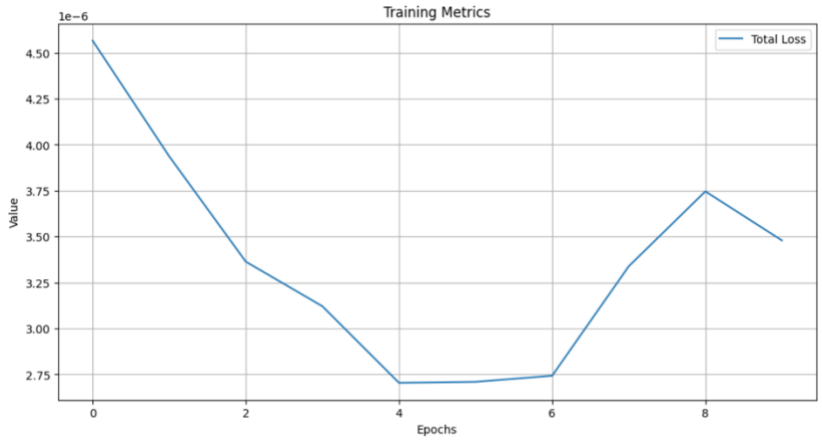


RetinaNet is a one-stage object detection framework that balances speed and accuracy by addressing the class imbalance problem inherent in object detection tasks. It introduces the Focal Loss, which focuses training on hard-to-detect objects while down-weighting easy examples. Unlike two-stage detectors like Faster R-CNN, RetinaNet directly predicts class

probabilities and bounding box offsets from dense anchor boxes, enabling faster inference. In this project, we fine-tuned a RetinaNet model with a ResNet-50 backbone pre-trained on ImageNet. The model was adapted to detect traffic elements such as vehicles, pedestrians, and traffic signals, annotated in COCO format. The network's anchor-based design ensures precise localization, making it a reliable choice for detecting small and large objects alike.

The training process involved fine-tuning the RetinaNet model using classification and Smooth L1 regression losses. Images were preprocessed using resizing and normalization, while bounding boxes were converted to RetinaNet's required format. An Adam optimizer and a learning rate scheduler were employed to optimize the model over 10 epochs. The dataset was split into training and validation sets, and evaluation was conducted using mean Average Precision (mAP) across multiple Intersection over Union (IoU) thresholds to measure detection accuracy. This approach ensured that the model achieved a balance of performance and computational efficiency, making RetinaNet suitable for detecting traffic elements in complex environments.

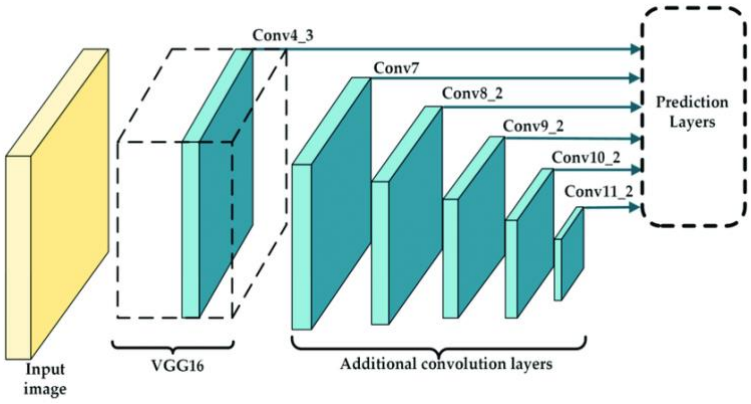
After training:-



5.1.4 Single Shot Detector

One well-liked object identification framework that is renowned for its effectiveness and quickness is the Single Shot Detector (SSD). In contrast to region-based techniques such as Faster R-CNN, SSD does object detection in a single network forward pass, obviating the requirement for region proposal stages. By mixing predictions for various scales and aspect ratios straight from feature maps produced by a deep convolutional network, like VGG16 or ResNet, it accomplishes this. For every point on the feature map, the SSD design employs predefined anchor boxes, sometimes referred to as default boxes, at various scales and

aspect ratios. On the basis of their Intersection over Union (IoU) overlap, each anchor box is paired with a ground truth object. Both the class probabilities and the anchor box offsets with respect to the ground truth objects are predicted by the network. SSD is a popular model in applications needing quick and precise object detection because of its design, which makes it especially good at identifying objects of different sizes while preserving real-time speed.

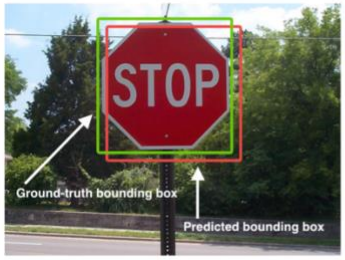


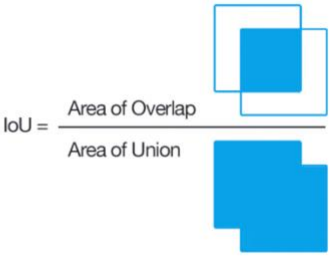
The SSD model is created using a pre-trained VGG16 backbone and customized with a new classification head to match the number of classes for the specific dataset (12 classes in this case). The model is moved to the GPU if available. An SGD optimizer with a cyclical learning rate scheduler is used to optimize the model's parameters. The training loop iterates over the dataset for a set number of epochs, during which the model performs forward and backward passes to minimize the classification and bounding box regression losses. The total loss is computed by weighing the individual losses, and gradient clipping is applied to prevent exploding gradients. The learning rate is adjusted dynamically using the cyclical scheduler to enhance training stability and convergence. After each epoch, the average loss is printed, and the model continues training until completion.

5.2 EVALUATION METRICS

5.2.1 IoU

Intersection over Union a metric used to evaluate how well an AI model can detect objects in an image. It's calculated by dividing the overlap between the predicted and ground truth bounding boxes by the union of the two. The IoU score ranges from 0 to 1, with 1 indicating a perfect overlap and 0 indicating no overlap. A score of 0.5 is typically considered good. Higher IoU indicates the predicted bounding box coordinates closely resembles the ground truth box coordinates. Commonly known as Jaccard Index.





5.2.2 mAP

Mean Average Precision (mAP) is a metric used to evaluate object detection models such as Fast R-CNN, YOLOv8, Mask R-CNN, etc. The mean of average precision (AP) values is calculated over recall values from 0 to 1. mAP formula is based on the following sub metrics: Confusion Matrix, Intersection over Union (IoU), Recall, Precision.

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k$$
$$AP_k = \text{the AP of class } k$$
$$n = \text{the number of classes}$$

Summary of steps to calculate:

1. Generate the prediction scores using the model.
2. Convert the prediction scores to class labels.
3. Calculate the confusion matrix—TP, FP, TN, FN.
4. Calculate the precision and recall metrics.
5. Calculate the area under the precision-recall curve.
6. Measure the average precision

5.3 EXPLAINABLE AI TECHNIQUES

5.3.1 Grad-CAM

Gradient-weighted Class Activation Mapping (Grad-CAM) [2] is a widely used visualization technique for interpreting convolutional neural networks (CNNs). It highlights the critical regions of an input image that influence the model's predictions. By generating class-discriminative localization heatmaps, Grad-CAM offers insights into how a model makes decisions without altering its architecture or compromising accuracy.

Grad-CAM computes the gradient of the predicted class score with respect to feature maps in the final convolutional layer of the CNN. These gradients are weighted and combined with the feature maps to produce a heatmap, known as the Class Activation Map (CAM), which highlights the important regions in the image for the specific prediction.

5.3.2 Saliency Maps

A saliency map [3] is an image that highlights the most relevant regions of an image for a machine learning model or what the human eye would focus on first. It's also known as a pixel attribution map. Saliency maps are a great tool to understand what convolutional layers are seeing in computer vision, allowing us to use these models in production in an informed manner.

A saliency map is computed by analyzing the gradients of a model's output score with respect to the input image, highlighting the most influential regions for a specific prediction. The input image is passed through the model, and the target score (e.g., for a class or detection) is selected. By backpropagating this score, gradients are obtained, representing how sensitive the score is to each pixel. The saliency map is derived by taking the absolute maximum gradient across the color channels, yielding a 2D heatmap that emphasizes regions with the highest impact on the prediction. This process provides insights into the model's focus without altering its architecture.

6. CHALLENGES

YoloV8: - One of the key challenges faced during the YOLO implementation was adapting the pre-trained YOLOv8 model to effectively work with the custom dataset, requiring careful tuning of hyperparameters and data preprocessing. Additionally, generating Grad-CAM and saliency maps for explainability proved complex due to the model's output structure and compatibility issues with visualization libraries. We encountered frequent errors, including mismatched tensor shapes and gradient computation failures, which required extensive debugging and modifications to ensure compatibility with YOLO's architecture. Despite these hurdles, we successfully achieved explainability through Grad-CAM and saliency maps and obtained a commendable mean Average Precision (mAP), demonstrating the model's strong detection performance and interpretability.

RetinaNet: - Implementing RetinaNet posed several complex challenges, particularly due to its reliance on COCO-style annotations and the requirement to convert and preprocess data into the correct formats. Generating a valid CSV annotation file and class mapping was cumbersome, as errors like malformed class IDs and mismatched bounding box formats repeatedly disrupted the workflow. Additionally, the integration of the fine-tuned RetinaNet model with custom evaluation pipelines, including the computation of metrics such as mean Average Precision (mAP), was fraught with compatibility issues. The model's predict function often failed due to discrepancies in input tensor shapes and bounding box formats, especially when handling non-standard image dimensions or dataset-specific peculiarities. Furthermore, attempting to implement explainability techniques, such as Grad-CAM and saliency maps, was complicated by RetinaNet's multi-head architecture, requiring intricate handling of its feature pyramid layers and anchor-based predictions.

Detection Transformer (DETR): The training of the DETR model had multiple challenges, including high and stable loss values, slow convergence, and difficulty achieving low loss. Despite fine-tuning across several stages, the loss remained very high, preventing meaningful progress. Initially, training the model without fine-tuning the backbone (ResNet50) yielded poor results, while fine-tuning the backbone significantly increased the training time. Despite extensive efforts, reducing the loss to an acceptable level proved difficult. Although the model and corresponding processor were saved after each stage, precision, recall, and Intersection over Union (IoU) metrics remained low throughout all stages.

Single Shot Detector (SSD): In the case of SSD, training for longer epochs might have resulted in better results, but the loss started very high early on and reduced extremely slowly. Despite using a Learning Scheduler and fine tuning the hyperparameters the loss starting loss was upwards of 15. The mAP, IoU did not cross 0.1 and ideally those metrics should be around 0.5 to be considered good or acceptable. We will continue training it and hopefully will have better results before presenting it to the class.

7. RESULTS AND ANALYSIS

7.1 IoU – Mean Intersection over Union

Model	Mean IoU
Faster R-CNN	0.84
YoloV8	0.69
RetinaNet	0.56
Single Shot Detector	0.07

7.2 IoU - Mean Intersection over Union per Class

Model / Class	Obstacles	Biker	Car	Pedestrian	trafficLight	trafficLight-Green	trafficLight-GreenLeft	trafficLight-Red	trafficLight-RedLeft	trafficLight-Yellow	trafficLight-YellowLeft	Truck
Faster R-CNN		0.78	0.86	0.77	0.83	0.76	0.81	0.8	0.81	0.73	0.89	0.87
YOLOv8		0.67	0.85	0.62	0.836	0.721	0.656	0.824	0.799	0.620	0.124	0.83
Single Shot Detector		0.1	0.1	0.05	0.1	0.05	0.05	0.05	0.05	0.05	0.05	0.1
RetinaNet		0.62	0.71	0.63	0.65	0.72	0.61	0.73	0.63	0.73	0.74	0.64

7.3 mAP – Mean Average Precision

Model	mAP
Faster R-CNN	0.78
Yolov8	0.688
RetinaNet	0.65
Single Shot Detector	0.09

7.4 mAP – Mean Average Precision per Class

Model / Class	Obstacles	Biker	Car	Pedestrian	trafficLight	trafficLight-Green	trafficLight-GreenLeft	trafficLight-Red	trafficLight-RedLeft	trafficLight-Yellow	trafficLight-YellowLeft	Truck
Faster R-CNN		0.8	0.91	0.87	0.91	0.8	0.88	0.85	0.82	0.77	0.2	0.91
YOLOv8	0.67	0.68	0.67	0.85	0.62	0.83	0.72	0.65	0.82	0.79	0.61	0.12
Single Shot Detector		0.1	0.2	0.1	0.1	0.05	0.05	0.05	0.05	0.05	0.05	0.1
RetinaNet	0.56	0.43	0.65	0.62	0.59	0.32	0.56	0.65	0.76	0.68	0.64	0.63

7.5 Model Training Time

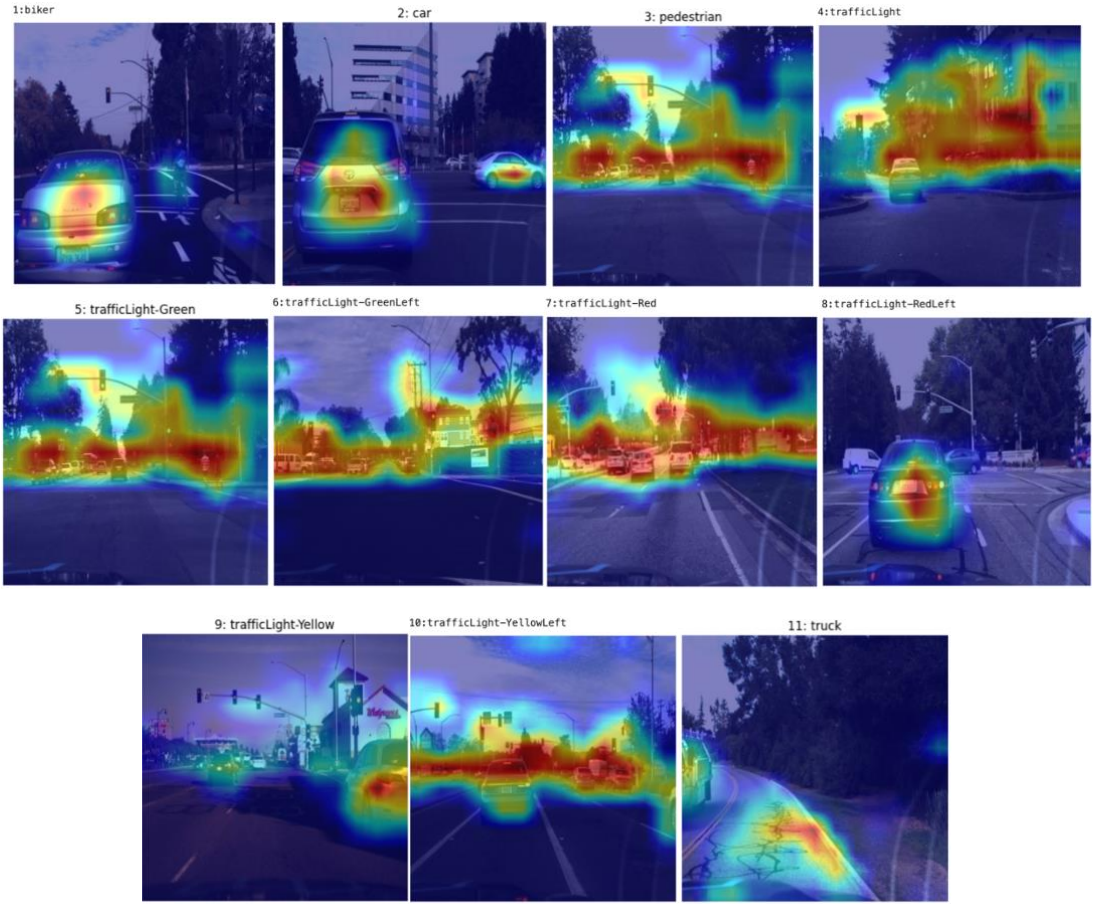
Model	Number of Epochs	Training Time (hours)
Faster R-CNN	12	10
Yolov8	20	5
RetinaNet	12	16
Single Shot Detector	12	1.2

7.6 Model Architecture and Complexity

Model	Complexity	Total Parameters	Trainable Parameters	Number of Layers	Memory Usage	Architecture
Faster R-CNN	High	41,350,411	41,128,011	189	Higher	Two Stage
YOLOv8	Low	31,157,200	30,157,200	168	Efficient	Single Stage
Single Shot Detector	Medium	25,082,528	25,043,808	80	Moderate	Single Stage
RetinaNet	Medium	35,430,992	35,377,872	152	Moderate	Single Stage

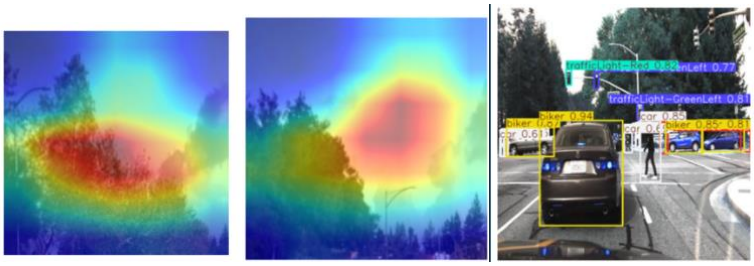
7.7 Grad-CAM for each Class

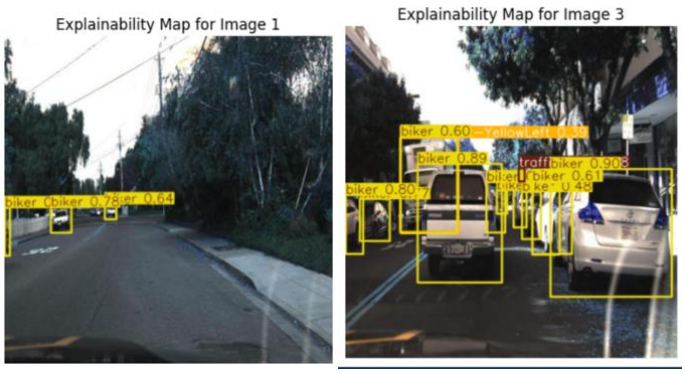
7.7.1 Explanations for Faster R-CNN



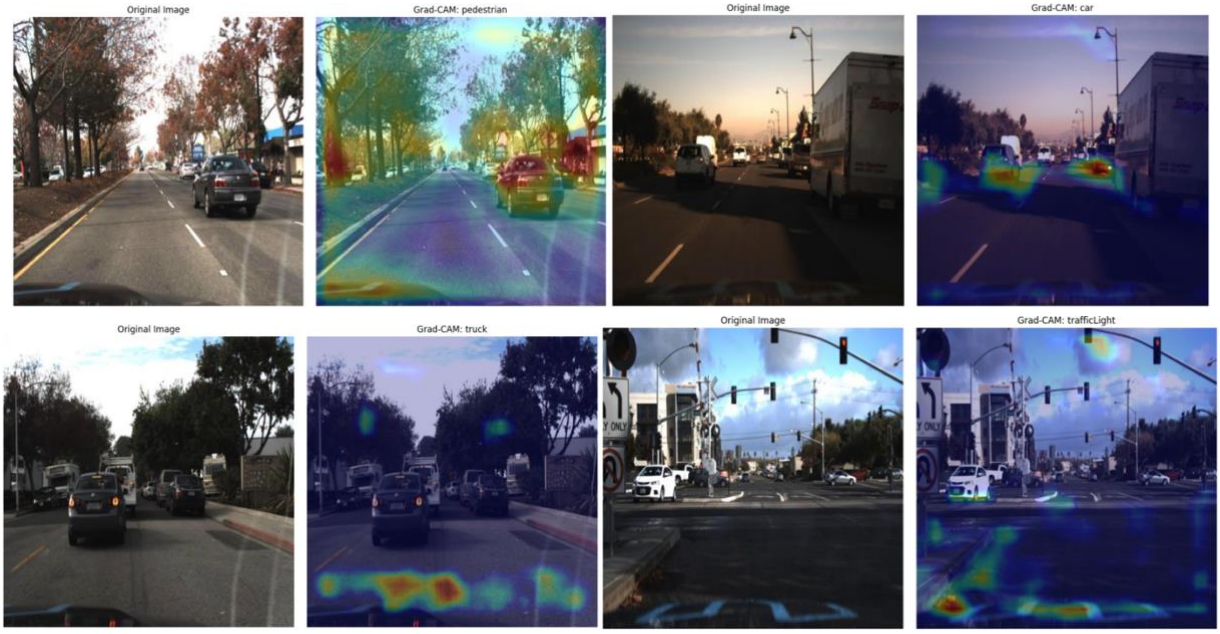
7.7.2 Grad-Cam for First Few Classes

Explanation for Yolov8



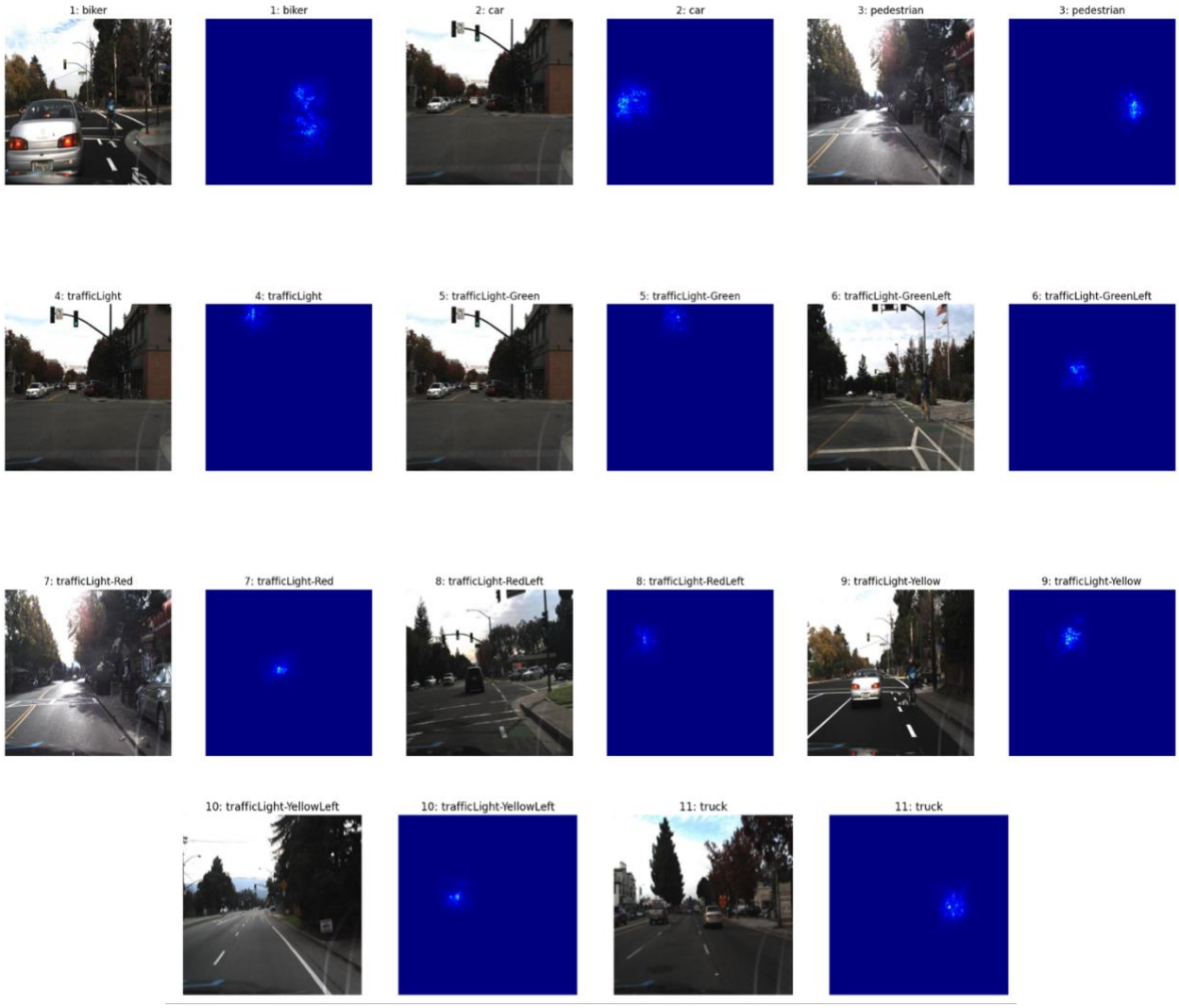


7.7.3 Single Shot Detector

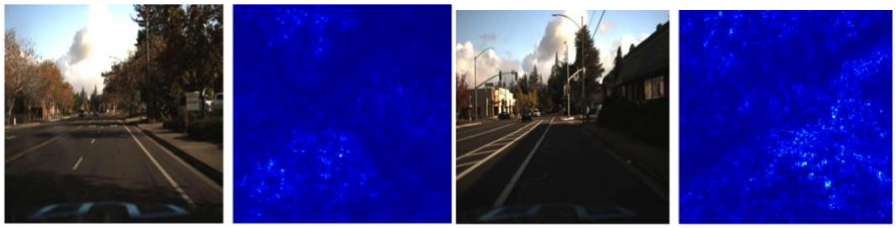


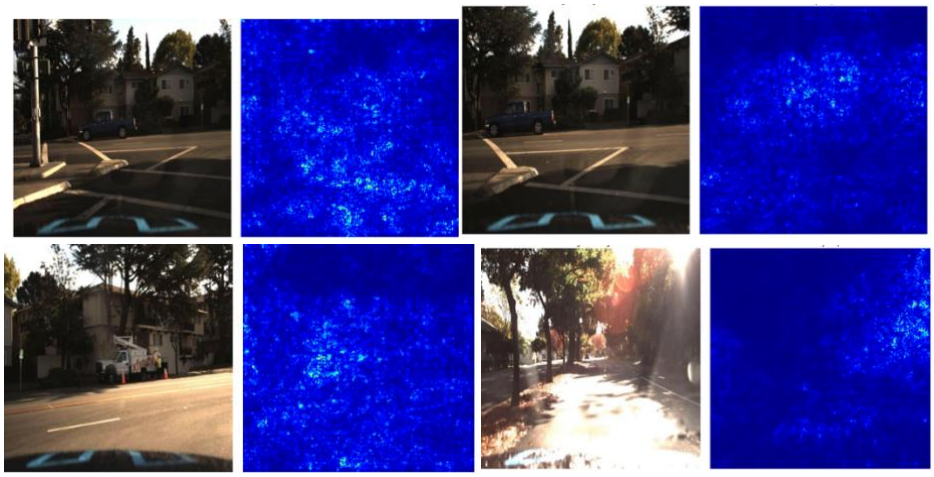
7.8 Saliency Maps for each Class

7.8.1 Faster R-CNN Explanations

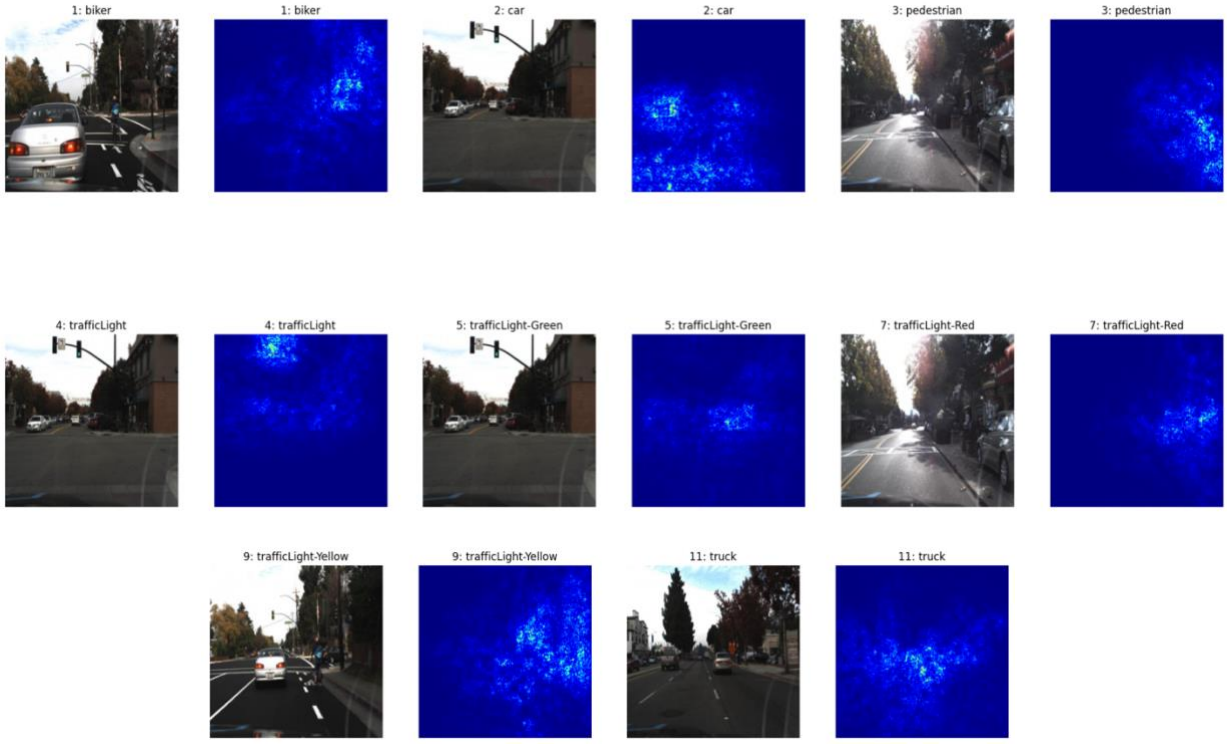


7.8.2 Saliency Map for First Few Classes (Yolov8)





7.8.3 Single Shot Detector



7.8 Comparative Analysis of Results

Model Complexity and Architecture

Faster R-CNN:

- Highest complexity with 41,350,411 total parameters
- Two-stage architecture with higher memory usage 189 layers

YOLOv8:

- Lowest complexity with 31,157,200 parameters
- Single-stage architecture with efficient memory usage 168 layers

RetinaNet:

- Medium complexity with 35,430,992 parameters
- Single-stage architecture with moderate memory usage 152 layers

Single Shot Detector:

- Medium complexity (exact parameter count not provided)
- Single-stage architecture with moderate memory usage

Accuracy and Performance

- Faster R-CNN has the highest mAP and IoU, given its complex architecture and higher parameter count
- YOLOv8 offers a good balance of accuracy and efficiency
- RetinaNet likely provides moderate performance
- Single Shot Detector has lower accuracy compared to the other models

Grad-CAM Explanation Analysis

Faster R-CNN:

- Shows precise focus on relevant areas for each class
- For vehicles (biker, car, truck), it concentrates on the body and wheels
- For pedestrians, it focuses on the lower part of the image where people are typically found
- For traffic lights, it highlights the upper portions of the image

YOLOv8:

- Demonstrates broader activation areas compared to Faster R-CNN

- Shows strong activation for the central object in the image
- Appears to consider more context around the main object

Single Shot Detector:

- Exhibits less focused and more scattered activations
- For pedestrians, it shows activation across a large portion of the image
- For trucks and traffic lights, the activations are more localized but still less precise than Faster R-CNN

Saliency Map Analysis

Faster R-CNN:

- Extremely focused, pinpoint attention maps
- Clear, singular activation points for each object class, minimal noise
- Precise localization of traffic lights and their states
- Strong correlation between attention and actual object location

YOLOv8:

- Broader, more distributed activation patterns
- Multiple attention points across the scene
- More contextual awareness in activation maps
- Shows activation clusters rather than single points
- Good balance between focus and scene understanding

Single Shot Detector:

- Most scattered and diffuse saliency patterns
- Less concentrated activation points
- Higher noise levels in attention maps
- Poor object-specific localization Inconsistent attention distribution

Complexity vs. Accuracy Trade-off:

- Faster R-CNN's high complexity seems to translate into more precise object localization and classification, as evidenced by its focused Grad-CAM visualizations.
- YOLOv8, despite lower complexity, appears to maintain good performance with its efficient architecture.

Overall Summary

Architectural Influence: The two-stage architecture of Faster R-CNN likely contributes to its ability to focus on specific object features, while single-stage models like YOLOv8 and SSD tend to consider broader context.

Feature Learning: Faster R-CNN's Grad-CAM results suggest it learns more discriminative features for each class, which may explain its presumed higher accuracy.

Efficiency vs. Precision: YOLOv8's lower complexity and efficient memory usage, combined with its broader but still relevant activations, indicate a good balance between speed and accuracy for real-time applications.

Model Limitations: The Single Shot Detector's less focused activations may explain its potentially lower accuracy, suggesting it might struggle with precise object localization.

In conclusion, the Grad-CAM visualizations provide valuable insights into how model complexity translates to feature learning and object detection capabilities. Faster R-CNN's higher complexity appears to result in more precise object focus, while YOLOv8 offers a compelling balance of efficiency and performance. The choice between these models would depend on the specific requirements of the application, weighing factors such as accuracy needs, computational resources, and real-time processing demands.

The saliency maps reinforce previous conclusions while providing deeper insights into each model's attention mechanisms. Faster R-CNN's precise saliency maps align with its high complexity and accuracy, while YOLOv8's balanced attention patterns support its position as a practical compromise between performance and efficiency. The Single Shot Detector's scattered saliency maps further confirm its limitations for practical applications.

8. CONCLUSIONS

8.1 TAKEAWAYS

1. Model Performance Spectrum:

- Faster R-CNN excels in precision and accuracy, particularly for critical object detection.
- YOLOv8 offers a balanced approach, combining good accuracy with efficient processing.
- Single Shot Detector (SSD) shows limitations in both precision and reliability for autonomous driving tasks.

2. Explainability Insights:

- Saliency maps and Grad-CAM visualizations provide crucial insights into model decision-making processes.
- Faster R-CNN demonstrates pinpoint accuracy in its attention maps, especially for traffic lights and vehicles.
- YOLOv8 shows broader, more contextual awareness in its attention patterns, beneficial for complex scenarios.

3. Real-world Applicability:

- YOLOv8's balance of speed and accuracy makes it more suitable for real-time autonomous driving applications.
- Faster R-CNN's high precision is valuable for specific safety-critical tasks but may be computationally intensive for full-time use.

4. Safety Considerations:

- Faster R-CNN's precise detection of traffic light states is crucial for safety-critical decisions. YOLOv8's broader context awareness aids in handling complex traffic scenarios more effectively.

In conclusion, a hybrid approach could be considered for future development, potentially leveraging Faster R-CNN's precision for specific safety-critical tasks in conjunction with YOLOv8's efficient general object detection. This combination could maximize both safety and performance in autonomous driving systems. The project underscores the importance of explainable AI in evaluating and selecting models for autonomous driving, providing valuable insights beyond mere performance metrics and enabling more informed decisions in model selection and system design.

8.2 APPLICATIONS

Possible Applications:

- Automotive Industry: Providing actionable insights for improving autonomous driving technologies, enhancing safety, and increasing reliability.
- AI Research and Development: Offering a methodological approach to evaluating and interpreting complex neural network architectures.
- Regulatory Compliance: Supporting the development of transparent AI systems that can meet stringent safety and accountability standards.

Our work contributes to:

- Increasing public confidence in autonomous vehicle technologies
- Advancing the field of explainable artificial intelligence
- Creating more transparent and trustworthy AI decision-making processes

9. REFERENCES

1. Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks
2. Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, Dhruv Batra, Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization
3. Karen Simonyan, Andrea Vedaldi, Andrew Zisserman, Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps
4. Marchisio, A., Martina, M., & Masera, G. (2021). Explainable Deep Neural Networks for Autonomous Driving: The Road to Transparency. IEEE Transactions on Vehicular Technology.