A study on YOLOv8 Object Detection for Advanced Driver-Assistance Systems

James Overmeyer
Electrical and Computer Engineering
University of Florida
Gainesville, FL, USA
jovermeyer@ufl.edu

***Abstract*—Object detection is an essential task in computer vision that is important to a wide range of applications. Image sear, autonomous systems, and surveillance are a few of the areas where object detection is utilized. Over the years, a variety of object detection deep learning-based algorithms have been proposed. Object detection systems may require different areas of focus, that is, some systems may prefer accuracy, whereas other systems may prefer speed. The balance of accuracy and speed is important when designing an object detection system, especially when choosing an object detection algorithm. One algorithm that has achieved significant improvements related to accuracy, speed, and ability of object detection is the YOLOv8 algorithm. Compared to classical object detection algorithms like R-CNN, YOLOv8 utilizes a single-stage object detection approach that uses a single CNN to produce bounding boxes and class probabilities from a single image input. This single-stage approach allows YOLOv8 to process images at up to 90 FPS but comes with slightly decreased accuracy especially for detection of small objects. This research project will examine the strengths and weaknesses of the YOLOv8 object detection algorithm. The YOLOv8 algorithm is utilized for object detection on the FLIR dataset. This FLIR dataset represents the use case of object detection for autonomous vehicles and robotics. A custom RISE algorithm was utilized for explainability of the YOLOv8 algorithm for infrared detection. The YOLOv8 algorithm demonstrated a variety of combinations of accuracy and speed depending on the YOLOv8 model size. The high accuracy and high speed of the YOLOv8 detection algorithm makes it suitable for utilization in self-driving applications.**

Keywords—object detection, YOLOv8, R-CNN, CNN, machine learning, deep learning.

I. Introduction

Object detection is a computer vision process that involves the identification and localization of objects in an image. Object detection has a range of applications that include robotics, autonomous vehicles, and surveillance. In use, object detection algorithms are utilized to automatically detect objects in an image or video and outputting a bounding box around the detected object and a class prediction for the object. The output of the object detection algorithm will then be utilized to trigger a response based on the application. For example, if an autonomous vehicle detects a person 10 meters in front of the vehicle, the detection of that person will cause the vehicle to brake.

These object detection algorithms usually consist of a single convolutional neural network or a series of convolutional neural networks. A neural network contains an input layer, hidden layers, and an output layer. A neural network with multiple hidden layers is considered a deep learning neural network. Each hidden layer acquires values based on the values of the previous layer, weights, biases, and an activation function. In convolutional neural networks convolutional layers utilize kernels that essentially act as filters across the spatial dimension of the input. When an input is passed into a trained convolutional neural network, certain neurons will 'fire' or activate when certain spatial features are present in the input. Pooling layers are utilized to reduce dimensionality by reducing the complexity of the representation. The weights and biases of the neurons are trained utilizing the backpropagation algorithm. Backpropagation, propagates the training error back to the input and changes the weights and biases of the neurons in the network to minimize an error function. Backpropagation is performed for a specified number of epochs or until convergence occurs [1,2].

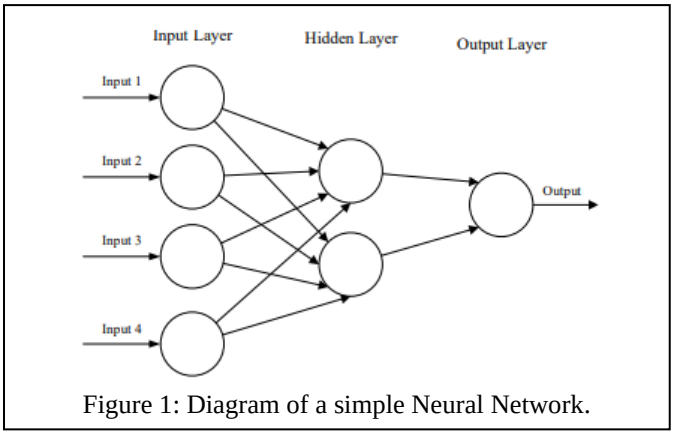

Figure 1: Diagram of a simple Neural Network.

The most common type of object detection algorithm is the region-based object detection. These region-based detection algorithms consist of a two-stage process, a proposal generation stage and an object classification stage. In the proposal generation stage, a sliding window with a variety of scales and aspect ratios is utilized across the image. Each sliding window size and positions is considered as a proposed region, and a prediction score is calculated predicting whether or not the proposed window contains an object. In the object classification stage, a deep neural network is applied to each proposal predicting whether or not the region contains a specific object. This two-stage process can be computationally expensive as tens of thousands of regions may be proposed for a given image, but

because of the large number of proposals the two-stage process is highly accurate and can detect small objects and occluded objects.

Faster R-CNN is one of the most popular region-based detection algorithms. The Faster R-CNN algorithm utilizes a Regional Proposal Network (RPN) to generate proposals, and

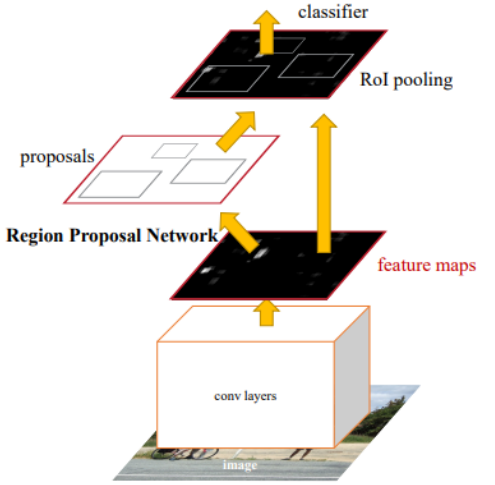

Figure 2: Diagram demonstrating shared computations for convolutions for Faster R-CNN.

the Fast R-CNN network to detect objects in the proposed regions. RPN is used to generate rectangular object proposals with an objectness score. A fully-convolutional network is utilized to generate these proposals, and both the RPN and the Fast R-CNN network share convolutional layers, and this shared computation allows for faster object detection. While Faster R-CNN has achieved state-of-the-art detection speeds compared to other region-based object detection algorithms of 17 FPS, 17 FPS is still insufficient for real-time object detection applications like autonomous vehicles and robots especially because optimization is made difficult by having to train the different components separately [3].

For applications that require real-time object detection higher frame rates are required, and this requirement is satisfied by single-stage object detection algorithms. The most advanced of which is the YOLOv8 algorithm. YOLOv8 is the eighth

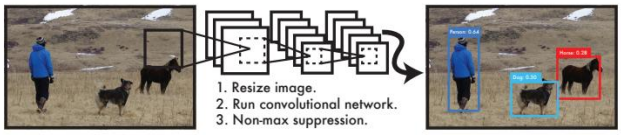

Figure 3: Diagram demonstrating single-stage detection using the YOLO algorithm.

iteration of the You Only Look Once architecture. In the YOLO algorithm, a single-convolutional network simultaneously predicts bounding boxes and class probabilities for those bounding boxes. The YOLO CNN utilizes 24 convolutional layers followed by 2 fully connected layers. The utilization of this single-layer detection method allows for real-time detection at speeds of up to 155 fps.

Generally, when it comes to accuracy, YOLO algorithms perform worse than two-stage detection algorithms like Faster R-CNN, performing about 8% lower. However, YOLO can also be utilized to enhance accuracy of two-stage detection algorithms. Combining YOLO with Faster R-CNN contributed to a 3% improvement in accuracy compared to only Faster R-CNN.

However, when it comes to real-time detection YOLO is unparalleled. YOLO algorithms achieved more than double the accuracy of other state-of-the-art real-time detection algorithms, and even achieved higher accuracies at higher frame rates.

The YOLO object detection algorithm has demonstrated immense high accuracy and high frame rates simultaneously, making it an ideal candidate for utilization in real-time object detection applications. This paper will demonstrate the real-time object detection capabilities of the eighth iteration of YOLO, YOLOv8, for thermal object detection for self-driving vehicles. This project will utilize the Teledyne FLIR infrared dataset to train and test the performance of a few different YOLOv8 model configurations. Additionally, a custom Randomized Input Sampling for Explanation (RISE) technique will be utilized to explain the detections of the YOLO algorithm [4,5].

II. Overview of YOLO

As previously mentioned, the YOLO algorithm is capable of achieving real-time detection speeds because it utilizes single-stage detection. This single-stage detection is achieved by unifying the components of object detection. The system divides the image into an S x S grid, and if an object falls into a specific grid cell, then that grid cell must detect the object. The grid cells each predict B bounding boxes with confidence scores. The confidence scores should be zero if no object exists in that cell or equal to the intersection over union, Figure 4. The prediction utilizes the intersection over union which is an accuracy measure between the predicted bounding box and the ground truth bounding box. Each grid cell predicts the C conditional class probabilities $Pr(Class_i|Object)$ conditioned on the cell containing an object. For testing the conditional class probabilities are multiplied to create a confidence score that encodes the confidence that a specific class appears in a box and how well the predicted box fits the object. Overall, a tensor with

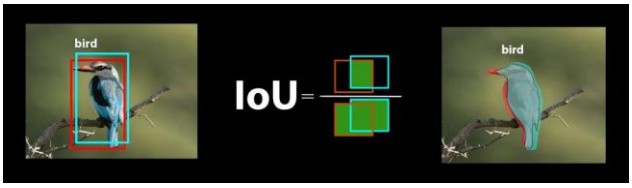

Figure 4: Intersection over Union calculation

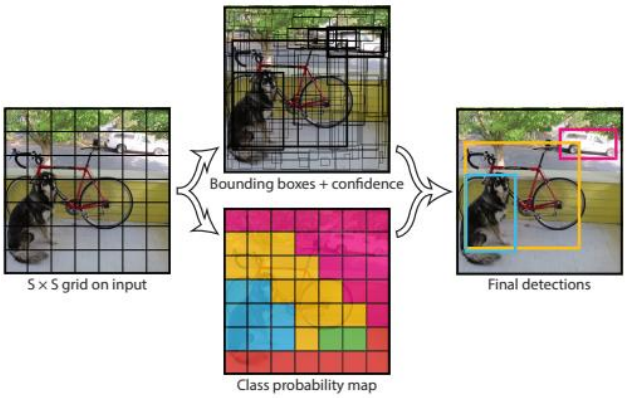

Figure 5: Framework for Detection

hape S × S × (B ∗ 5 + C) is the output of the system, as shown in Figure 5.

$$\Pr(Class_i|Object) * \Pr(Object) * IOU_{pred}^{truth} = \Pr(Class_i) * IOU_{pred}^{truth} \quad (1)$$

The training framework is implemented using a 24 convolutional layer neural network with 2 fully connected layers. The convolutional layers extract features of the images, and the fully connected layers predict the output probabilities and coordinates as the S × S × (B ∗ 5 + C) tensor.

For detection 4 additional convolutional layers and 2 additional fully-connected layers are added to the network, as shown in Figure 6. The final layer of the network utilizes a linear activation function, whereas the other layers use a leaky ReLU. For an input x, leaky ReLU will output 0.1x if x is negative and x if x is positive.

To optimize the network, the sum-squared error is optimized. Optimization of the sum-squared error weighs localization and classification errors equally. This is not ideal, so loss parameters are included to decrease the loss from boxes that don't contain objects.

Object detection algorithms are typically judged by their mean average precision (mAP) and framerate. The mAP is an accuracy measure that is calculated based on the area under the curve of a precision-recall curve. The mAP for object detection is typically calculated using a specific intersection over union threshold.

For real-time object detection, YOLO has demonstrated outstanding performance. Achieving an mAP of 63.4 for real-time detection at 45 FPS. Even more, the Fast YOLO network, with only 9 convolutional layers, has achieved a mAP of 52.7 at 155 FPS. The best real-time detector aside from YOLO, Deep MultiBox, only achieves an mAP of 16 for 100 FPS and 26.1 for 30 FPS, making YOLO and Fast YOLO both more than 2x better than their closest competitors.

For overall object detection, Faster R-CNN achieves a 73.2 mAP, 10 higher than YOLO, but YOLO has a 6x higher framerate. YOLO can also be combined with Fast R-CNN to achieve higher accuracies for less than real-time detection. Alone, Fast R-CNN achieves an mAP of 71.8, but combined with YOLO achieves an mAP of 75. This gain of 3.2 mAP can be attributed to the two detection algorithms making different types of mistakes, so performance is boosted when combined.

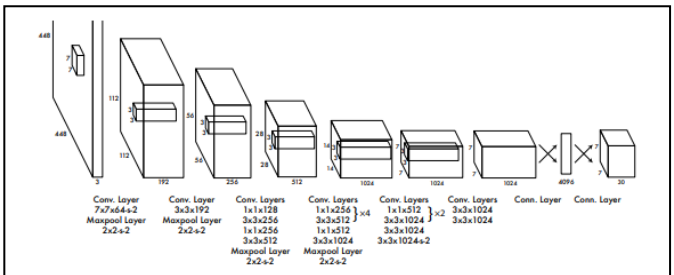

Figure 6: Neural Network for YOLO detection

III. Related Work

R-CNN and single shot detectors have been one of the state-of-the-art algorithms for object detection for many years. Akshatha, et al demonstrated the performance of Faster R-CNN and SSD on a thermal dataset [6]. The thermal dataset being utilized was the AAU PD T dataset which contains thermal images of pedestrians acquired using an aerial camera in complex outdoor environments. This work demonstrated an mAP of 55.7% for Faster R-CNN, but only achieved 8 FPS in detection. SSD exhibited a higher framerate at 44 FPS, but only achieved an mAP of 25.9%. This work demonstrates that these object detection algorithms have some shortcomings when it comes to real-time object detection. Additionally, the dataset utilized in this work only contained labeled pedestrians, and thus the algorithms only had to detect one class type. If multiple class types were included in the dataset, the speed and mAP would both be lower for both algorithms. This work shows that in general, FRCNN and SSD are insufficient for a complex real-time object detection algorithm such as autonomous driver assistance system. These ADAS systems require high framerates above 60 FPS, and must have object detection capable of detecting a variety of objects from cars to pedestrians to traffic lights. The literature demonstrates that these higher accuracy object detection algorithms have not yet been implemented for real-time detection, and that real time detection using these algorithms may not be possible without sacrificing a massive amount of accuracy.

A lot of work has been performed demonstrating the utilization of YOLO for object detection, but very few projects have studied YOLO for the specific application of thermal

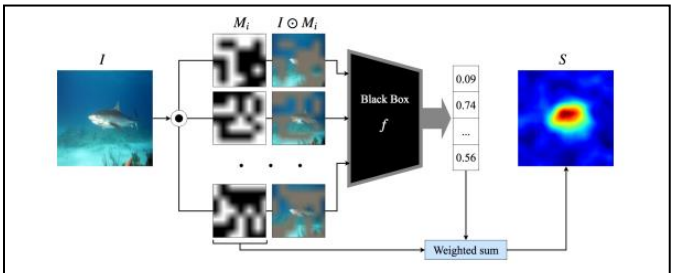

Figure 7: RISE detection algorithm

imaging. Kos, et al. utilized the YOLOv3 network to perform human detection using thermal imaging in different weather conditions [7]. Utilizing an IoU threshold of 50%, the YOLOv3 network trained on a thermal dataset achieved a 30% AP,

whereas the YOLOv3 network only trained on the Microsoft COCO dataset achieved an AP of 7%. This work demonstrated that training the object detection network on a thermal dataset is vital to achieving high performance in object detection. Jiang, et al evaluated and compared the performance of different YOLO networks on a UAV thermal dataset [8]. This project concluded that the YOLOv3 framework had the highest mAP of 88.69% and fastest speed of 50 FPS, however the YOLOv5-s model had the smallest size of 14 MB which may be beneficial for applications where small amounts of storage may be available. Kristo, et al evaluated a variety of object detection networks for thermal object detection in difficult weather conditions [9]. They established that a YOLOv3 network trained on a thermal dataset of 3000 images achieved an AP of 97.93% for human detection in all weather conditions, and YOLOv3 trained only on the COCO dataset achieved 19.63% AP for human detection in all weather conditions. These previous projects have demonstrated the potential of YOLO object detection networks for high accuracy thermal object detection. However, prior projects have failed to utilize any explainability criteria to explain what is occurring during YOLO object detection of thermal objects.

IV. RISE

Explainability is crucial to an object detection model. While evaluating a model on a validation dataset can demonstrate good performance metrics. This only demonstrates that the model learned to detect the scenarios in the validation dataset, but the validation dataset may not be representative of all scenarios. The necessity of model explainability led to the development of random input sampling for explanation (RISE). The RISE algorithm creates heatmaps that explain black box object detection algorithms. A number of random masks are applied to an individual input image, and the masked inputs are all passed through the black box algorithm. The output contains bounding boxes with class labels and confidence values. For a given class, the masked images are weighted and summed according to the predicted confidence. Using this weighted sum, a heat map is generated for each class showing what components of the image are most important to making a classification. RISE may not fully explain what is occurring in the black box object detection algorithm, but due to the sparsity of black box algorithm explainability is the best explainability method available as it is capable of explaining region importance to a human observer [10, 11].

V. Experiments and Evaluation

To prove the efficacy of the YOLOv8 algorithm for advanced driver-assistance systems, a thermal dataset from the Teledyne FLIR website was utilized. Thermal datasets are important for ADAS applications because thermal cameras can provide information in any weather or light conditions, making it suitable for detection objects like other vehicles or pedestrians even when they are not visible to the human eye. The dataset consists of 26,442 images and a JSON file containing the bounding boxes and class labels for 520,000 annotations. This dataset was converted to the YOLO format by creating a text file

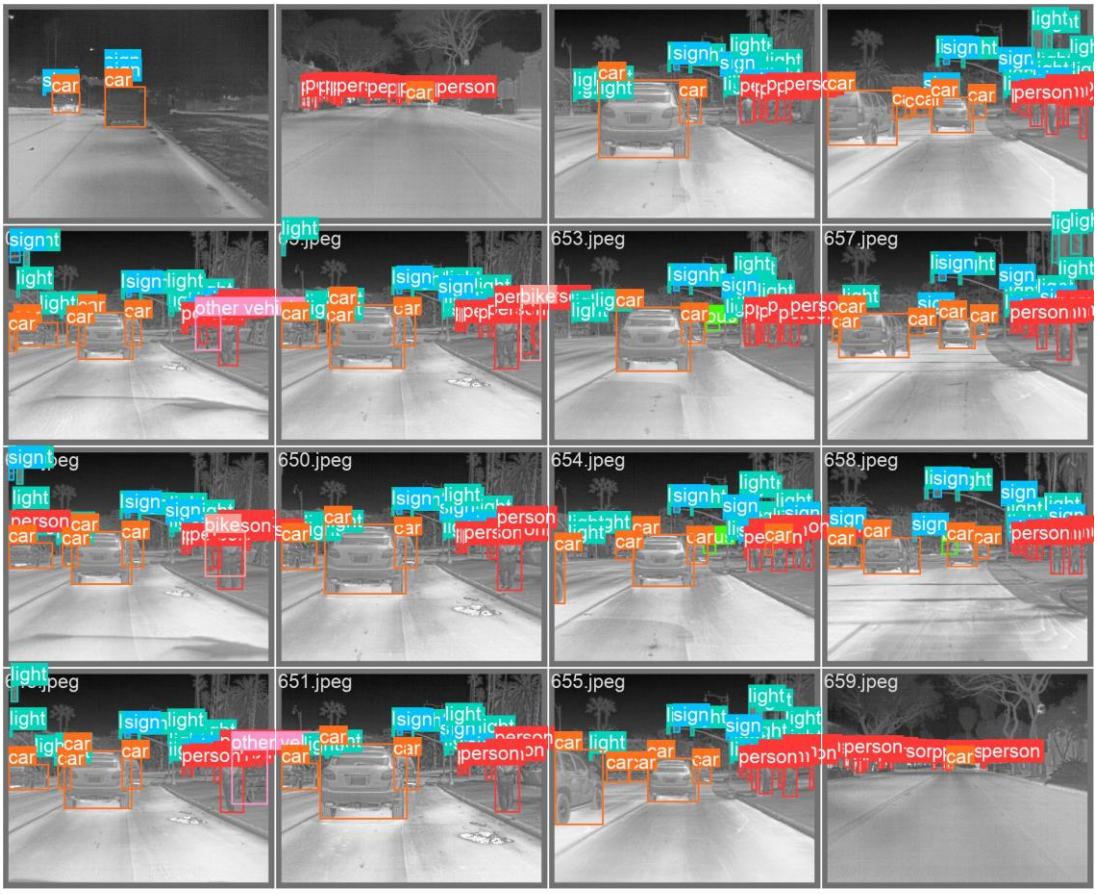

Figure 8: Original images with labels

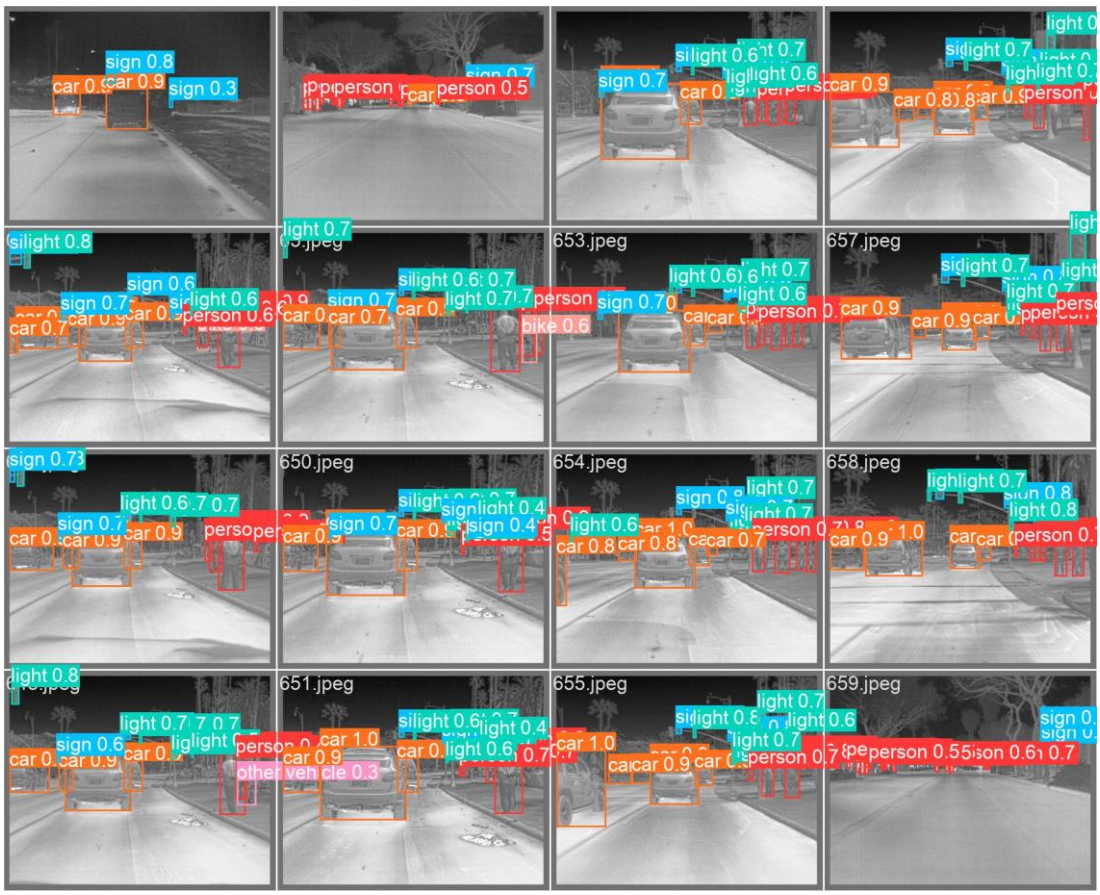

Figure 9: YOLOv8 generated labels.

for each image. The text file has the same name as the image file and contains the bounding box and class information for each annotation in the given image. Each bounding box and class combination is a separate row in the text file.

The YOLOv8 library has 5 pre-trained models available, nano, small, medium, large, and extra-large with 3.2M, 11.2M, 25.9M, 43.7M, and 68.2 parameters, respectively. These models come pre-trained on the COCO dataset [12]. For the current experiments the nano, small, and medium models were implemented for object detection in the FLIR dataset using an A100 GPU on the University of Florida Hipergator network. The nano and small models both have 225 total convolutional layers, and the medium has 295 total convolutional layers. Each model was trained with 20 epochs and a batch size of 4 and with 40 epochs and a batch size of 16. These epochs and batch sizes were chosen to evaluate the difference in training accuracy between many epochs and a large batch size and few epochs and a small batch size. A total of 6 models were trained using the combinations.

Model	mAP50	Time
Epoch-20 Batch-4 N	0.3889	2 Hrs
Epoch-40 Batch-16 N	0.4038	2.5 Hrs
Epoch-20 Batch-4 S	0.4434	2.25 Hrs
Epoch-40 Batch-16 S	0.4467	2.5 Hrs
Epoch-20 Batch-4 M	0.4621	2.75
Epoch-40 Batch-16 N	0.5147	3.25 Hrs

Table 1: Results of training for YOLOv8 models

10,742 images were utilized for training each model. The longest training time was for the 40-epoch medium model, and it lasted 3.25 hours. The training time for the medium model was significantly longer than the training times for the smaller models due to the additional network layers. High levels of mAP were achieved on a validation set for all model sizes as shown in Table 1. The 40-epoch medium model achieved the highest mAP of 5.147.

The trained models were then utilized for prediction on a test set of FLIR images. Examples of the YOLOv8 predictions are shown in Figure 9, with the ground truth labels shown in Figure 8. For predictions using the nano, small, and medium models, 90, 85, and 63 FPS were achieved, respectively. The mAP and FPS combinations demonstrate the immense potential of YOLOv8 for real-time object detection applications.

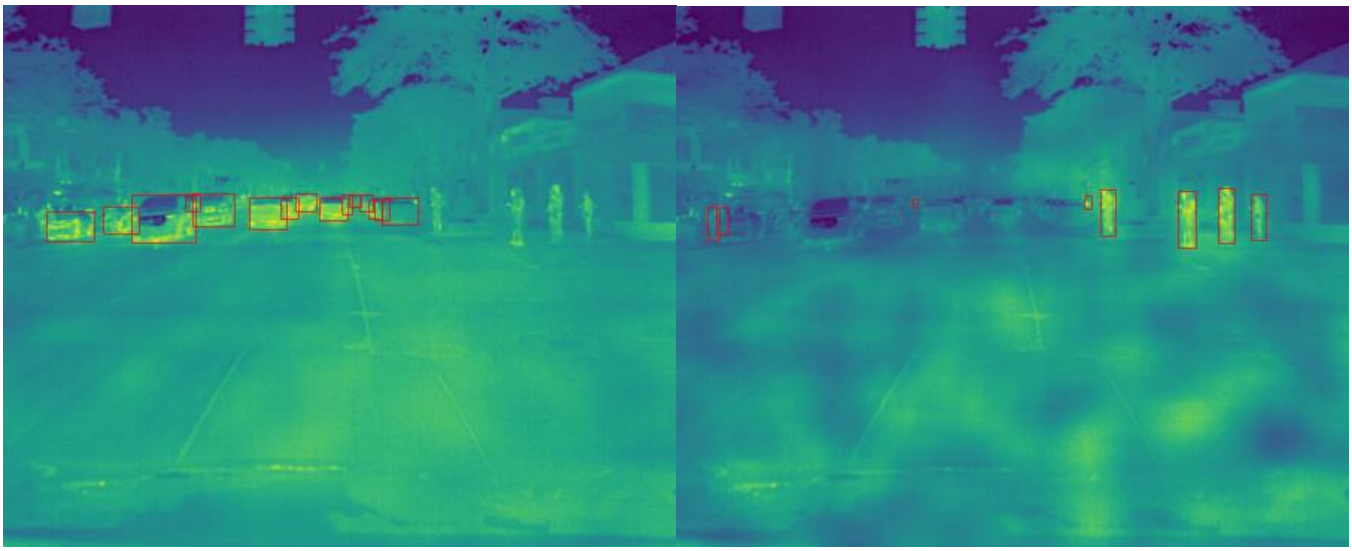

Figure 10: RISE explainability for cars, left, and people right.

While the speeds achieved make real-time object detection possible. It is still difficult for the YOLOv8 models to detect small objects. Figure 7 has a few examples of some small objects that were not detected by the model. For autonomous vehicle applications, these objects may often be pedestrians, street signs, or streetlights. The system does not struggle with detecting larger objects like cars and bikes. The difficulty that the YOLO models have in detecting small objects is the primary reason that YOLO networks cannot achieve the mAP that is demonstrated by R-CNN and other non-real-time object detection models. As previously mentioned, R-CNN has demonstrated a mAP as high as 72 for less than real-time detection.

Figures 8 and 9 demonstrate the ground truth labels and bounding boxes and the predicted labels and bounding boxes for the YOLOv8 FLIR detection. From these images it can be observed that the YOLOv8 algorithm is capable of detecting a variety of objects with a variety of shapes and sizes with high accuracy. However, it is clear from these images that the YOLOv8 algorithm still struggles to detect small objects. This is an important trade-off when making considerations for a real-time detection algorithm. While detecting objects quickly is important for real-time decision making, being able to detect small objects such as far away people and vehicles is important, so that advanced driver assistance systems can recognize potential hazards in advance. ADAS systems will often utilize a variety of sensors and detection algorithms simultaneously to develop a wholistic approach to automated decision making.

It is well known that YOLO is a black box object detection model, so it is difficult to understand how the model is making object detections. Because of this the utilization of some explainability criteria is vital to demonstrate why the model is making some detections and not making others. Good criteria for explainbility are difficult to produce for such an algorithm, and a lot of research has been performed on new approaches like LIME and RISE for model explainability.

For this study a complicated custom algorithm was utilized to implement RISE explainability on a few of the images from the test dataset. Given a test image, the selected model was utilized to perform detection. For each unique detected class in the image, 1000 random masks were generated to mask the image. These masks were created by randomly dimming individual pixels within the image to 0. The masked images were then passed through the same detection model. If the class was detected, then the masked image was added to an output image array. After 1000 iterations for the selected class, a heatmap of the summed class detection was generated. The generated heatmaps appear bright in regions that were present for many detections.

RISE heatmap generation is a lengthy process, as YOLO object detection is performed for 1000 iterations for every image. While this method generates localizations that are useful for human explanation, the generated heatmaps may not actually be capturing how a decision is made. However, the results clearly indicate that the object that is being detected, itself, is the most important aspect for a detection to occur. Figure 9, demonstrates this for people and for vehicles. The RISE heat maps also show how the algorithm fails to detect small objects, as smaller people in the background are detected when no masking occurs but are frequently not detected when slightly masked. The RISE heat maps also demonstrate that some contextual information may contribute to the detections as the heatmap appears brighter around the tires of a vehicle than around the windshield, so tires may contribute more to vehicle detections than windshields, etc.

VI. Summary and Conclusions

During this project, the YOLOv8 framework was successfully utilized to perform real-time object detection on a FLIR dataset. The results demonstrated that a mean average precision of 0.5 or more is achievable for framerates as high as 60 FPS. This result demonstrates that YOLOv8 is suitable for inclusion in ADAS systems. ADAS systems require high accuracy and high-speed object detection simultaneously. Furthermore, YOLOv8 detection can be combined with other sensors and algorithms to achieve a wholistic model sensing approach to ADAS systems. RISE explainability was also implemented for the first time to explain the YOLOv8

algorithms object detection on a FLIR dataset. The results of the RISE algorithm show that YOLOv8 is good at detecting medium to large size objects, but may struggle with detecting small objects. The RISE results also show that contextual information may play a key role in YOLOv8 making accurate detections. The importance of contextual information to detection may be a key area of interest in object detection algorithms in areas that require high accuracy and speed of detection like ADAS. If the system is placed in a relatively new seeming environment, then it may struggle to perform accurately.

The YOLO family of algorithms has evolved very quickly over the past five years and will continue to develop making it suitable for more and more applications, and this work demonstrates one area where it may continue to be involved.

REFERENCES

[1] S.S.A. Zaida, et al, "A survey of modern deep learning based object detection models," Digital Signal Processing, 2022, vol 126.

[2] K. Oshea, et al. "An Introduction to Convolutional Neural Networks," 2015.

[3] R. Girshick. "Fast R-CNN," Proceedings of IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1440-1448.

[4] J, Redmon, et al, "You Only Look Once: Unified, Real-Time Object Detection," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp 779-788.

[5] J. Terven, et al, "A Comprehensive Review of YOLO: From YOLOv1 to YOLOv8 and Beyond," ACM Computing Services, 2023.

[6] K.R. Akshatha, et al. "Human Detection in Aerial Thermal Images Using Faster R-CNN and SSD Algorithms." Electronics. 2022, 11, 1151.

[7] M Ivasic-Kos, et al,. "Human detection in thermal imaging using YOLO," ICCTA '19: Proceedings of the 2019 5th International Conference on Computer and Technology Applications. 2019, pp. 20-24.

[8] C. Jiang, et al, "Object detection from UAV thermal infrared images and videos using YOLO models," International Journal of Applied Earth Observations and Geoinformation, 2022.

[9] M. Kristo, et al. "Thermal Object Detection in Difficult Weather Conditions Using YOLO," IEEE Access. Volume 8. 2020.

[10] V. Petsiuk, et al, "RISE: Randomized Input Sampling for Explanation of Black-box Models." Computer Vision and Pattern Recognition (CVPR), 2018.

[11] V. Petsiuk, et al, "Black-box Explanation of Object Detectors via Saliency Maps," Computer Vision and Pattern Recognition (CVPR), 2021. Oral.

[12] T. Lin, et al, "Microsoft COCO: Common Objects in Context," European CCV, 2014, pp. 740-755.