DEVELOPMENT OF DETECTION AND TRACKING SYSTEMS FOR AUTONOMOUS VEHICLES USING MACHINE LEARNING

A Thesis

Presented to

the Faculty of the Elmer R. Smith College of Business and Technology

Morehead State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

---

by

Tyler Ward

April 25, 2023

Accepted by the faculty of the College of Business and Technology, Morehead State University, in partial fulfillment of the requirements for the Master of Science degree.

_____
Dr. Sherif Rashad
Director of Thesis

Master's Committee: _____, Chair
Dr. Sherif Rashad

_____
Dr. Ahmad Zargari

_____
Dr. Heba Elgazzar

_____
Date

DEVELOPMENT OF DETECTION AND TRACKING SYSTEMS FOR AUTONOMOUS VEHICLES USING MACHINE LEARNING

Tyler Ward
Morehead State University, 2023

Director of Thesis: _____
Dr. Sherif Rashad

Human activity recognition and prediction systems are crucial to the safety of autonomous vehicles. While much research has been conducted to improve these systems, very little has been done to address the important task of differentiating between adult and child pedestrians. Failure to correctly identify the type of pedestrian can lead to accidents.

In this thesis, a novel multiple object tracking system for autonomous vehicles is proposed that overcomes the challenges of differentiating between adult and child pedestrians. To increase the system's robustness, it is also capable of identifying and tracking 51 different animal types that are commonly encountered on roads around the world. The proposed system uses modern machine learning methods for object detection and tracking to identify the type of pedestrian or animal, and also measure various characteristics of their behavior, such as speed and trajectory. Experimental results indicate effectiveness in accomplishing these tasks, demonstrating the potential of the multiple object tracking system to improve the safety and performance of autonomous vehicles.

Accepted by: _____, Chair
Dr. Sherif Rashad

_____
Dr. Ahmad Zargari

_____
Dr. Heba Elgazzar

Table of Contents

CHAPTER 1: INTRODUCTION

The first section of this chapter provides a brief introduction to autonomous vehicles and their interactions with the environment around them. The second section of this chapter discusses the application of machine learning in these vehicles, specifically on the common computer vision tasks of object detection and tracking.

1.1 Autonomous Vehicles

Though autonomous vehicles (AVs) have long been present in science fiction media, high expenditures have kept manufacturers from meaningful AV production until fairly recently (Fagnant, 2015). However, as AV technology becomes cheaper and more advanced, there have been widespread pushes from traditional car manufacturers such as BMW, Mercedes-Benz, Nissan, and General Motors to ramp up production of AVs, partly to ward off competition from new entrants into the field such as Google (Bagloee, 2016). It is estimated that in 2021, the global market demand for AVs was 51.6 thousand units, with that number expected to grow by 53.6% by 2030 (Autonomous, n.d.).

Despite the increasing prevalence of AVs on roads around the world, public opinion is contentious. Surveys show that the majority of U.S. drivers are worried about sharing the road with these vehicles, do not want to ride in one, and favor policies restricting their use (Smith, 2019). The reason for this public weariness likely comes from the fact that traditional vehicles have been on roads for over a hundred years, allowing the public to become acclimated to them and learn how to react around them. Such acclimation does not yet exist for AVs, signaling the importance for them to function well enough that they are indistinguishable from vehicles driven by humans.

An AV's architecture consists of three layers: the perception layer, the decision layer, and the action layer (Wang, 2020). The perception layer collects data from exterior sensors

(Ilas, 2013), which is processed in the decision layer to make decisions and pass information to the action layer (Ha, 2020). The action layer controls actions that mimic a human driver (Lee, 2020). Errors in any of these layers can lead to accidents.

Studies show that the majority of accidents involving AVs are caused by third parties such as pedestrians (Favarò, 2017). The perception and decision-making capabilities of the vehicle are most likely to be affected by these parties (Wang, 2020; Sarmento, 2017). This demonstrates the necessity for AVs to be able to adapt to different pedestrian behaviors.

For example, children are less likely to pay attention when crossing a road (Lee, 2020), are less capable of judging the speed of oncoming traffic (Clay, 1995), and are more likely to behave unpredictably than adults (Holland, 2007). To address this, a machine learning model for detecting and classifying pedestrians according to whether they are an adult or child is presented in this thesis, so that their movements can be accurately tracked and predicted by AVs to avoid accidents.

1.2 Machine Learning

Belonging to the encompassing field of artificial intelligence, machine learning (ML) is an increasingly important technology in the modern world. ML algorithms seek to teach computers how to perform certain tasks and make predictions and decisions at or above the level of humans. Deep learning (DL) algorithms seek to further increase the level of knowledge a computer can obtain and the type of activities they can perform. They do this by trying to computationally simulate how the human brain operates. ML has many applications, including computer vision.

1.2.1 Computer Vision

Computer vision (CV) is an application of ML that focuses on extracting information from a scene when presented with image or video input. This input can be considered as a discrete array of numbers representing brightness or color values at a discrete grid of points

in the image plane. This grid of points is more commonly referred to as pixels (Rosenfield, 1988).

CV techniques are extremely important to modern AVs, specifically in their perception capabilities. In terms of perception, CV aids the performance of AVs in multiple ways, including vision-based driver assistance systems and environment perception tasks such as lane detection, traffic sign/light recognition, and vehicle tracking (Wang, 2020; Ilas, 2013; Janai, 2020). There are many applications of CV, but generally, CV tasks can be classified as either object detection or object tracking tasks.

1.2.1.1 Object Detection

Object detection (OD) is a CV task that deals with detecting objects such as humans, animals, or cars in digital images or videos (Zou, 2023). OD has been employed for many tasks, such as autonomous driving, robot vision, and video surveillance (Zou, 2023). OD algorithms have come a long way since the early days of hand-crafted features and sliding window techniques, with the most recent DL-based methods achieving state-of-the-art performance on a variety of datasets. As research in CV increases, there is a growing interest in developing more efficient and accurate object detection algorithms, with a focus on real-time applications and novel network architectures.

Traditional methods of performing OD include Viola-Jones detectors, histogram of oriented gradients (HOG) detectors, and deformable part-based models (DPM) (Zou, 2023). Viola-Jones detectors were first introduced in 2001 to quickly discard background regions and focus on object-like regions, operating based on what was then a novel image representation technique called the “integral image”, a learning algorithm based on AdaBoost that selects critical visual features, and a cascading methods for combining classifiers (Viola, 2001). The HOG approach was proposed in 2005 and outperformed existing feature sets for human detection using fine-scale gradients, fine orientation binning, relatively coarse spatial

binning, and high-quality local contrast normalization in overlapping descriptor blocks (Dalal, 2005). DPM was developed as an extension of HOG, using a margin-sensitive approach for data mining hard negative examples with a formalism called latent SVM (Felzenszwalb, 2008).

Since 2012, most of the research surrounding OD techniques has been based on convolutional neural networks (CNN), in either one or two stages (Zou, 2023). Two-stage CNN approaches include regions with CNN features (R-CNN), spatial pyramid pooling networks (SPP-net), Fast R-CNN, Faster R-CNN, and feature pyramid networks (FPN). The most popular one-stage approach today is You Only Look Once (YOLO), proposed in 2015.

When it was proposed, R-CNN broke through the plateau in OD research by using high-capacity CNNs to bottom-up region proposals and localize and segment objects, while using supervised pre-training and domain-specific fine-tuning to improve performance (Girshick, 2014). The first improvement on the R-CNN method was SPP-net, which implemented spatial pyramid pooling into a CNN to remove the requirement for a fixed-size input image, which had been negatively impacting OD performance up to that point (He, 2015). The next improvement to R-CNN came in the form of Fast R-CNN in 2015, with much faster training and testing time, as well as increased OD accuracy (Girshick, 2015). Faster R-CNN improved the speed and accuracy even further and shared full-image convolutional features with the detection network, enabling nearly cost-free region proposals (Ren, 2017). YOLO, one of the most popular and accurate OD methods available today, was a new approach to OD, and reframed OD as a regression problem to spatially separate bounding boxes and associated class probabilities (Redmon, 2016).

1.2.1.2 Object Tracking

Object tracking, specifically multiple object tracking (MOT) is an important task in CV research, as it provides an extension on OD that can enable real-world applications of

these methods. For the use case of pedestrian detection for AVs, simply detecting the presence of a pedestrian would not be useful to the AV in avoiding an accident, but when the detection is combined with tracking capabilities, the AV can learn how to react to the pedestrian's behavior. MOT algorithms can generally be classified according to their initialization method, processing mode, and type of output (Luo, 2021). Each of these areas has two general classes that the MOT models can fall into. For the initialization method, an MOT model can either be detection-based or detection-free, for processing mode, the tracking can occur either online or offline, and for type of output, the model can be either stochastic or deterministic (Luo, 2021).

Detection-based tracking first detects objects and then links them to trajectories, while detection-free tracking requires manual initialization of fixed number of objects in the first frame, then localizes them in subsequent frames (Luo, 2021). In online tracking, the image sequence is handled in a stepwise manner, while offline tracking uses batches of frames to process the data (Luo, 2021). Stochastic tracking methods have varying results, whereas the results from deterministic tracking algorithms remains constant (Luo, 2021).

The general architecture of MOT models can be given as containing: An appearance model, a motion model, an interaction model, an exclusion model, occlusion handling, and inference capabilities (Luo, 2021). Modern appearance models are based on visual representation, which describes an object based on its features, and statistical measuring, which deals with the affinity between two observations (Luo, 2021). Motion models can handle either linear motion or non-linear motion, with linear motion models typically capturing an object's dynamic behavior, and non-linear motion models explain the dynamics of the object (Luo, 2021). Types of interaction models include social force models, where each object is considered to be dependent on other objects and environmental factors, and crowd motion pattern models which are designed to track individual objects within a crowd

(Luo, 2021). There are two types of exclusion models: detection-level exclusion modeling and trajectory-level exclusion modeling. Occlusion handling is done in either a part-to-whole strategy, where it is assumed that part of an object is still visible when an occlusion happens, a hypothesize-and-test strategy, where proposals are hypothesized and tested according to the observations at hand, or a buffer-and-recover strategy, which remembers states of objects of before occlusion (Luo, 2021). Inferences can be either probabilistic or deterministic, where probabilistic inference represents the state of objects as a distribution with uncertainty, and deterministic inferences aims to find the maximum a posteriori solution to MOT (Luo, 2021).

1.2 Research Goals & Objectives for Detection and Tracking Systems

This research focuses on the use of ML algorithms to create robust OD and OT systems for AVs. Due to the lack of meaningful research in the area of differentiating between adult and child pedestrians, this research work aims to create an OD and OT system that is capable of not only detecting adult and child pedestrians, but also tracking them through a scene, measuring characteristics about their behavior such as their estimated speed and predicted trajectory. In addition, in order to make the system more practical for real-world use, the system is expanded past adults and children, to also consider 51 classes of animals that are commonly encountered on roads around the world. Twelve main objectives were considered during the creation of this system.

- Creation of a custom dataset containing images of adult and child pedestrians.
- Development of a custom ML-based model that is capable of detecting and differentiating between adult and child pedestrians.
- Implementation of a ML-based OT algorithm to track the detected pedestrians.
- Development of a mathematical model for accurately estimating the speed of the detected pedestrians.

- Implementation of ML-based regression models for predicting the trajectory of detected pedestrians.
- Test the pedestrian detection and tracking systems on real-world scenarios.
- Creation of a custom dataset containing images of 51 different commonly encountered animals on the road around the world.
- Development of a custom ML-based model that is capable of differentiating and detecting each of the 51 animals.
- Implementation of a ML-based OT algorithm to track the detected animals.
- Implementing the created mathematical model to estimate the speed of the detected animals.
- Implementing the chose ML-based regression model to predict the trajectory of the detected animals.
- Test the animal detection and tracking systems on real-world scenarios.

CHAPTER II: LITERATURE REVIEW

The first section of this chapter provides a review of different systems focused on the differentiation between adult and child pedestrians. The second section provides a review of different systems for detecting animals on the road.

2.1 Adult vs. Child Pedestrian Detection and Tracking Systems

The struggle for AVs to properly detect a child in the road as well as it can an adult in the same situation is well documented. In a study comparing 33 OD methods for pedestrian detection on the commonly used INRIA dataset (Dalal, 2005) and the Caltech Pedestrian Detection Benchmark (Dollar, 2009), it was found that in 27 of the 33 methods, the miss rates for child pedestrians were much higher, with all of the 24 best performing models showcasing this reduced detection capability (Brandão, 2019). This represents a definitive age bias in the OD algorithms used by modern AVs that needs to be addressed.

One of the most common methods for differentiating between adult and child pedestrians is to detect the head and body separately, then use relative size measurements to make the prediction (Ince, 2014; Ince, 2015; Ince, 2017; Reyes-Garcia, 2019). This approach makes use of the Haar cascade method proposed by Paul Viola and Michael Jones in their seminal work on OD (Viola, 2001). A major limitation of this method is that it relies on the height of the pedestrian, which is difficult to estimate in digital images (Ince, 2014; Ince, 2015). This can lead to misclassification of shorter adults or taller children.

Even though accurate estimation of object height based on digital images is a challenge, there have been studies that have used this approach to create OD systems that differentiate between adults and children. A 2021 study presented an approach based on human body morphology and the YOLO algorithm to measure the height of a detected person to determine if they are an adult or child (Sedgh-Gooya, 2021). This approach was found to be effective in identifying whether a person was an adult or child in a variety of different circumstances,

including if the person was sitting, had their back to the camera, were bent over, or were covered by an article of clothing (Sedgh-Gooya, 2021).

Another common technique for differentiating between adult and child pedestrians is the use of so-called “regions of interest” within an image. This is a very similar approach to the “integral image” in the Haar cascade method. A 2017 study used this “region of interest” approach combined with HOG and support vector machines (SVM) to identify adult and child pedestrians within a scene (He, 2017). A similar approach was employed in another study that divided input images into sections and compared them to other images in order to decide whether a pedestrian was an adult or a child (Balbuzanov, 2019).

Aside from the ratio and region-based approaches that have been discussed, another way to handle differentiation between adults and children is to use appearance descriptors rather than relative measurements. This approach involves detecting the entire figure of the pedestrian, rather than making separate detections for the head and body. The Haar cascade method is decades old, and modern object detection methods, such as those based on CNNs, often use appearance descriptors as the basis for their class predictions (Han, 2015; Simo-Serra, 2015; Zagoruyko, 2015; Balntas, 2016; Heo, 2021; Kogure, 2022; Sharma, 2022).

CNNs are commonly used for image processing tasks such as pattern recognition. They consist of three main types of layers: convolutional layers, pooling layers, and fully connected layers. These layers transform the input image and produce class scores that can be used for classification and regression (O’Shea, 2015). In contrast, the Haar cascade method uses an “integral image” to evaluate the aspects of the original image that coincide with the elements to be detected and predict their classes. This is done by continuously eliminating sections of the image that are considered to be background (Viola, 2001). Studies show that CNN predictions are more accurate than those given by Haar cascade (Andrie Asmara, 2021).

Computer vision is a rapidly evolving field in computer science, and the YOLO architecture is currently one of the most popular, robust, and accurate CNN-based models. Several studies have applied YOLO to adult and child pedestrian classification, with some (Manikandan, 2019; Lin, 2022) using a head to body ratio scheme similar to (Ince, 2014; Ince, 2015; Ince, 2017; Reyes-Garcia, 2019) and others considering the whole body of the pedestrian (Heo, 2021). However, these studies are outdated because they use older versions of YOLO, and their classification accuracy and robustness can be improved. One study combines the object detection process with other tasks such as lane detection and object tracking but does not include important features for a robust autonomous vehicle system such as speed estimation and trajectory prediction (Heo, 2021).

OT is a natural extension of OD methods, as it allows computers to track detected objects across a given scene (Soleimanitaleb, 2022). Whereas YOLO is a prominent object detection method, Simple Online and Realtime Tracking with a Deep Association Metric *(*DeepSORT), an extension of the Simple Online and Realtime Tracking *(*SORT) algorithm proposed in (Bewley, 2016), is one of the most popular, robust, and accurate multiple object tracking models. DeepSORT uses an offline pre-training stage with a deep association metric and appearance information to reduce identity switches and computational complexity compared to the original SORT algorithm (Wojke, 2017). YOLO and DeepSORT methods have been combined by many for various MOT tasks, including pedestrian tracking (Mohideen Meera Sha, 2021).

Alternative methods for implementing OT into pedestrian detection models have been discussed. One study used CV techniques to analyze correlated, scale invariant locomotion properties to classify different styles of walking, which was then used to differentiate children from adult walkers based on relative stride length and stride frequency (Davis, 2001). Another study proposed a novel ML algorithm to detect moving pedestrians in real-time from a

stationary camera based on eigenflow, which are the eigenvectors derived from applying principal component analysis (PCA) to the optimal flow of moving objects (Goel, 2007).

Overall, there have been a number of research works that apply YOLO-based models to the subject of adult and child pedestrian detection. However, these studies are largely outdated, not robust, or do not implement OT into their systems. For this reason, we propose a pedestrian detection model based on the latest version of YOLO, YOLOv8, and combine the output of that model with the DeepSORT OT algorithm. This system is then expanded with a custom mathematical model for speed estimation, and ML-based regression models to predict the trajectory of the pedestrian.

2.2 Animal Detection and Tracking Systems

Traffic incidents involving animals have long been an issue, especially as there are more and more vehicles traversing roads all over the world. High animal mortality rates are one of the most visible negative impacts of roads and traffic flows on nature (Jaarsma, 2006). This problem has gotten more attention as the development of AVs has increased, and the unpredictability of animals moving into the path of an AV has been used by researchers as evidence that there still needs to be some element of human control in these vehicles (Szénási, 2020). Despite this, there has been much research into how to improve the capability of AVs to detect different types of animals in the road.

Many different methods have been applied to animal-in-road detection, but the research area of most interest to this thesis is the application of DL to this problem. Many studies have employed DL for animal detection (Matuska, 2016; Prabhakar, 2017; Sharma, 2017; Yudin, 2019; Hans, 2020; Protopapadakis, 2020; Levering, 2021; Santhanam, 2021; Mowen, 2022; Kahlon, 2023) to great effect. Some studies focused on only detecting a single type of animal in the road, such as cows (Sharma, 2017; Kahlon, 2023) and deer (Hans, 2020). Other studies focus on the detection of multiple animal types (Matuska, 2016;

Prabhakar, 2017; Yudin, 2019), and some studies focus on the detection of accidents that have already occurred that involved animals (Levering, 2021).

As with human pedestrian detection, the HOG technique has found application in the area of animal detection. A 2016 study trained a ML model using LBP-AdaBoost to detect animals up to 200 meters away from a car, and then combined this model with a HOG-SVM approach to identify only moose (Matuska, 2016). Another study employed HOG and the Haar cascade method to detect the presence of cows in the road in India and found that the HOG method performed better than the Haar cascade, with an accuracy of 82.5% compared to 78.1% (Sharma, 2017).

By far the most common approach to performing animal-in-road detection is the applications of CNNs. One study utilized a CNN to detect unsigned physical road incidents, including those caused by animals (Levering, 2021), and another used a CNN to detect animals in the road and trigger an alert 3 seconds before the point of impact to make the driver of the vehicle aware of the danger (Santhanam, 2021). A 2D-CNN was employed in another study with thermal image input to determine the risk imposed by a certain animal in a specific pose to vehicles at night (Mowen, 2023).

A two-class CNN was applied to differentiate a deer from its background in (Hans, 2020). A multi-label CNN was used to identify animals, debris, road defects, fire, fog, flooded areas, and humans in (Protopapadakis, 2020). A region-based CNN was used for the detection and classification of on-road obstacles such as vehicles, pedestrians, and animals in (Prabhakar, 2017).

YOLO-based models have also been applied to animal-in-road detection. A study in 2019 compared the YOLOv3 model to RetinaNet R-50-FPN, Faster R-CNN R-50-FPN, and Cascade R-CNN R-50-FPN and found that YOLOv3 demonstrated the highest accuracy on classifying animals from ten classes (Yudin, 2019). Another study used YOLOv4 to achieve

95.10% accuracy to identify the status of cattle on the road (Kahlon, 2023). Several of the aforementioned studies have combined their OD models with OT approaches as well (Matuska, 2016; Prabhakar, 2017; Sharma, 2017).

In the next chapter, a method for animal-in-the road detection and tracking is proposed. The system will be able to identify and track 51 different types of animals that are commonly encountered in roads around the world. Like the adult and child pedestrian detection and tracking system, this system will be based on the YOLOv8 OD model and the DeepSORT OT model and supplemented with the speed estimation model and the ML regression model for trajectory prediction.

CHAPTER III: PROPOSED METHODOLOGIES

3.1 Research Methodology for Pedestrian Detection and Tracking System

Figure 1 shows the block diagram of the proposed aged-based multiple object detection and tracking system. It starts with the collection of the data, followed by the detection of objects within the images. Once the objects are detected, they are classified. Human subjects will be classified based on their predicted class as either an adult or a child. Also, as a result of the detection and classification, the objects are given unique identifiers, so that they and their characteristics can be tracked throughout a scene. A customized version of the YOLOv8 algorithm is used for the object detection system.

For the tracking system, DeepSORT is used. This algorithm is capable of tracking each uniquely identified object through a scene, reidentifying them if they exit and then reenter a scene. Using this tracking system, the ID and starting and ending x and y positions are output to a separate file. This position history can then be used to estimate the speed and trajectory of the objects. A diagram showing how YOLO and DeepSORT work together is shown in Figure 2.

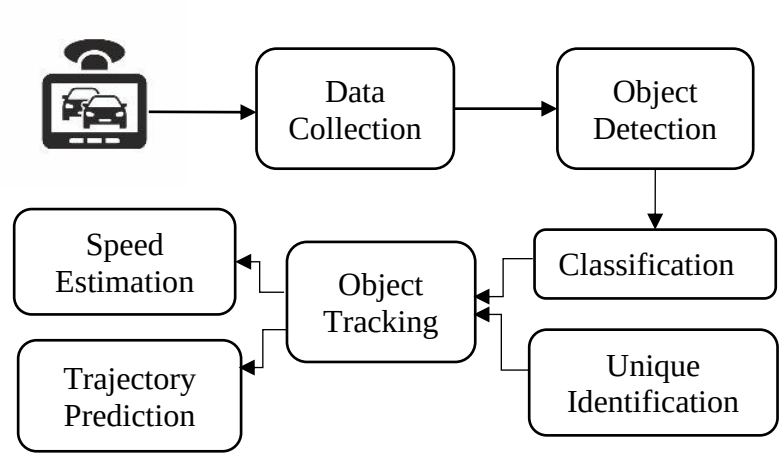

Figure 1 The block diagram of the proposed pedestrian detection and tracking system.

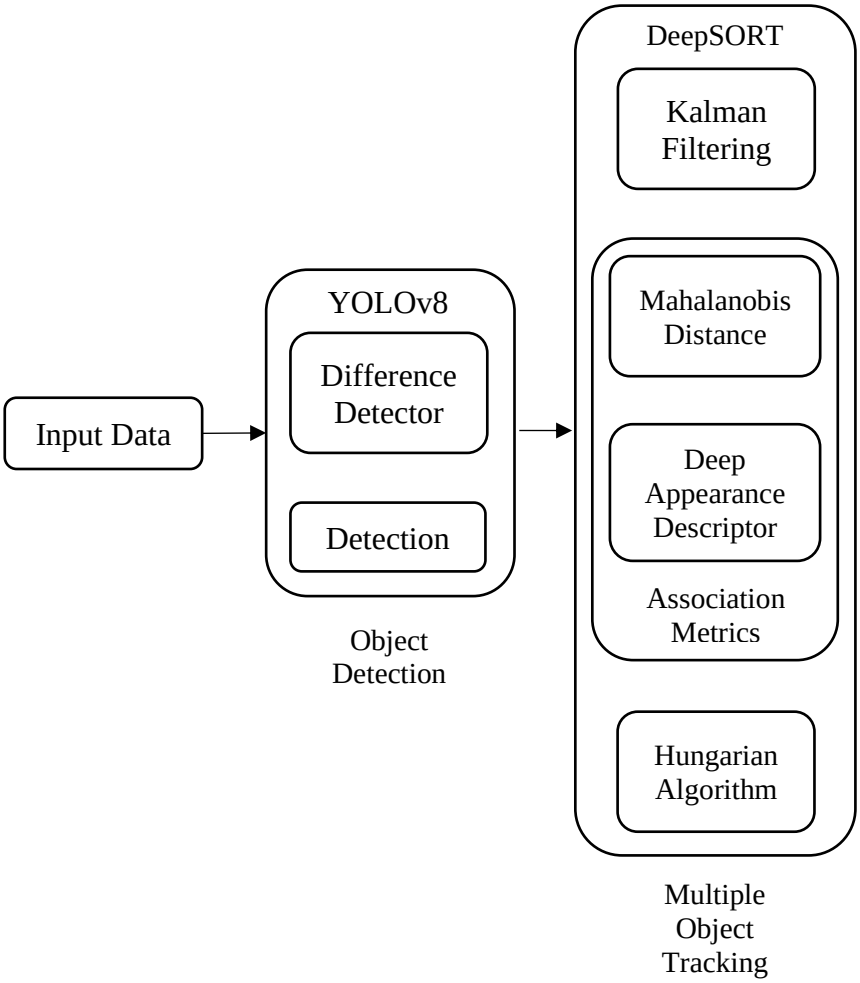

Figure 2 Diagram showing the combined process of YOLOv8 and DeepSORT.

3.1.1 Data Collection for Adult and Child Pedestrian Detection

Many datasets for AVs are publicly available, such as Berkely DeepDrive (Yu, 2020), Oxford Robotcar (Maddern, 2016; Barnes, 2020), and ApolloScape (Huang, 2018), but there are very few datasets that are adequate for the training of custom object detection models of pedestrians. Of the ones that do exist, it is even harder to find adequate samples for both adults and children crossing roadways together or separately. Because of this, a custom dataset was developed for this research project.

Image samples were taken from open-source websites, with the focus being on mixtures of adults and children crossing the road in various circumstances. In some cases, other notable objects, such as vehicles, were present in the scene, and in other cases the focus was solely on the pedestrians. Because of the intensive computation processing challenges of

large datasets, the curation of the custom set revolved around finding high-quality images, so that the overall number of samples could be lower but maintain good scalability on larger testing datasets.

The dataset used in this research was compiled using an online tool called Roboflow (Dwyer, 2022), and consisted of 150 high-definition images, that were an average size of 0.17 megapixels. When classified by median width and median height, there are 100 medium images, 25 large images, and 25 jumbo images. There were 634 total annotations across the images, with an average annotation per image of 4.2 across the two classes. The breakdown of annotations was 451 adults and 183 children, representing a slight under-representation of the children class.

Figure 3 shows the class balance of the pedestrian dataset. Figure 4 shows the dimension insights of the pedestrian dataset, including the size distribution and the aspect ratio distribution. Figure 5 shows a comparison between the heatmaps of the annotations of the adult and child classes.

Figure 3 The class balances in the pedestrian dataset.

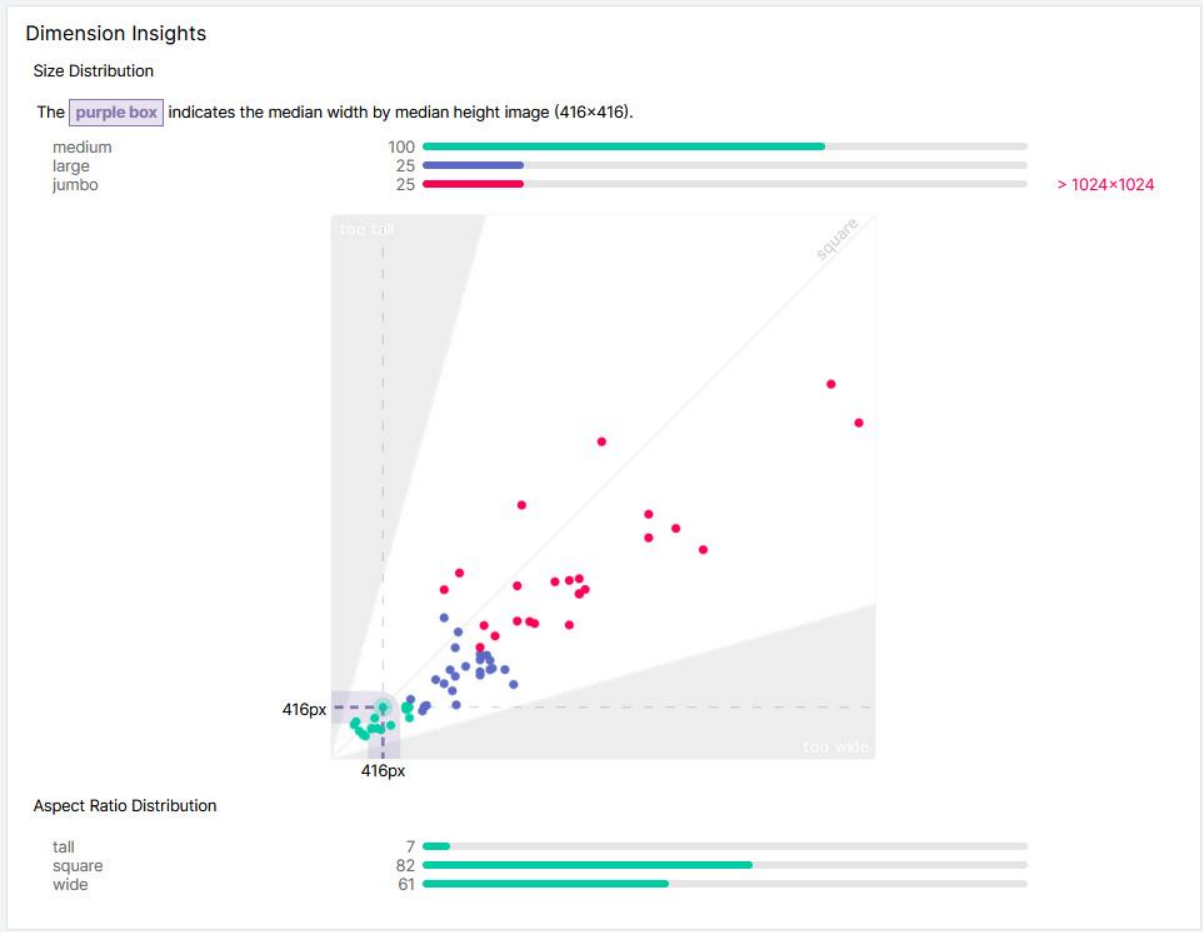

Figure 4 Dimension insights of the pedestrian dataset.

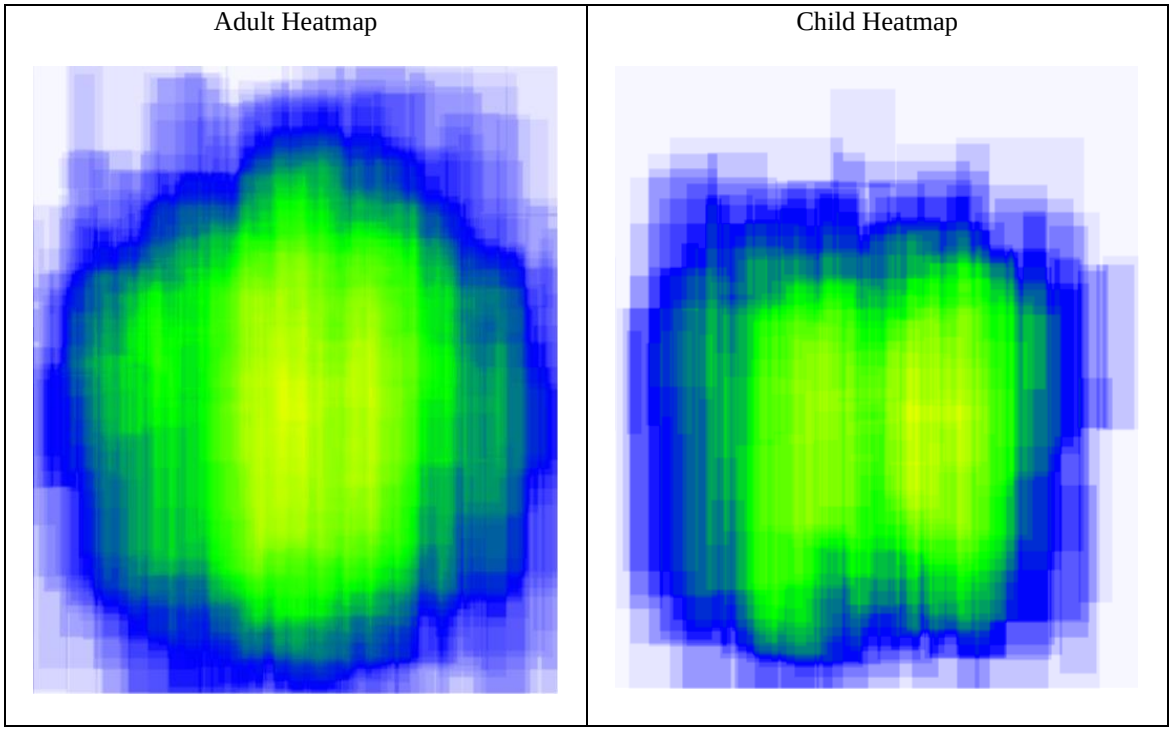

Figure 5 Comparison of the heatmaps between the adult and child classes.

3.1.2 Data Collection for Animal Detection and Tracking

As with the data collection process for the adult and child pedestrian detection and tracking system, a custom dataset was built for the proposed animal detection and tracking system using images collected from open-source websites and compiled with Roboflow. There are many datasets available that focus on enabling OD for animals, but many of these datasets are not as robust as the proposed dataset. The proposed dataset presented in this thesis consists of 51 classes of animals that are commonly encountered on roads around the world, as identified by (Knutson, 2006). All of the animals that are present in the dataset can be seen in Figure 6.

The animal dataset consists of 510 high-definition images, that were an average size of 0.93 megapixels. When classified by median width and median height, there are four small images, 116 medium images, 154 large images, and 236 jumbo images. Figure 6 shows the class balances of the animal dataset. Figure 7 shows the dimension insights of the animal dataset, including the size distribution and the aspect ratio distribution.

Figure 6 The class balances in the animal dataset.

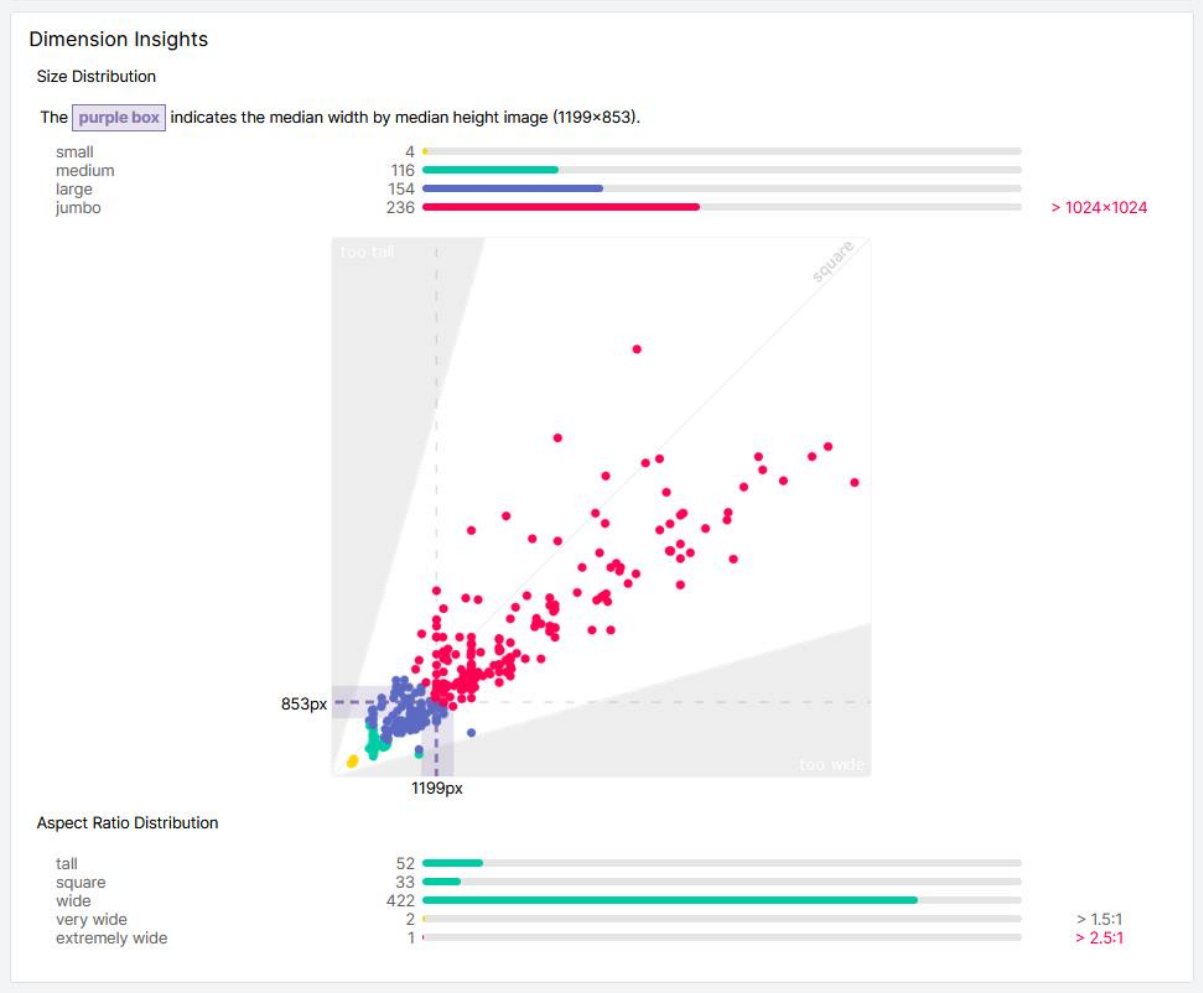

Figure 7 Dimension insights of the animal dataset.

3.1.3 Data Preprocessing

When an image is captured, it contains metadata that specifies the orientation in which the image should be displayed in relation to the arrangements of pixels on the disk. This metadata is saved in an exchangeable image file format (EXIF) orientation field and aids image encoding during capture. By following the directives given by the metadata, cameras can efficiently sample data from their sensors without creating undesirable artifacts. As a result, regardless of whether the camera is in portrait or landscape mode, most cameras save image pixels in the same manner.

Nevertheless, when a program displaying the images is unaware of the metadata and fails to follow the EXIF orientation, problems can occur. This is a common mistake that

easily occurs and is one of the most commonly encountered bugs in CV projects. An example scenario of this problem in action is if images with differing pixel configurations, such as (x, y) vs. (y, x) are saved on a disk, a user may train a ML model with inaccurate information without realizing it, leading to reduced model performance.

To avoid this issue, the images in the dataset undergo an auto-orientation process during the data preprocessing phase. This step will ensure that the ordering of pixels is uniform across all of the images. Studies show that data preprocessing goes a long way towards fixing any problems and is an important step when preparing to work with data in any capacity (Famili, 1997). Roboflow offers many different forms of data preprocessing techniques, and a comparison of their various effects on the pedestrian detection and tracking systems proposed in this research is given in a later section of this thesis.

3.1.4 Data Augmentation

As with data preprocessing, data augmentation is another process that can be undertaken prior to training a given ML model. Especially with deep CNN-based CV tasks, good generalization of the model can be challenging (Shorten, 2019). The reason that generalization is important to the model is because if a model has poor generalization, overfitting can occur, when can learn to a high training and validation error (Shorten, 2019). Data augmentation is a very powerful method that often reduces overfitting when applied to a model (Shorten, 2019).

For the purposes of this research, one data augmentation technique was used: mosaic augmentation. Roboflow enables many different data augmentation techniques, and this was determined to be the most beneficial to improving the proposed model. A comparison of all of the augmentation techniques provided by Roboflow and their various effects on the pedestrian detection and tracking systems proposed in this research is given in a later section of this thesis.

The mosaic data augmentation technique integrates four source images into a single output image (Hao, 2020). The simulation of random cropping of input images while keeping the relative size of items in relation to the overall image helps OD models deal with instances where object occlusion or translation has occurred (Hao, 2020). Another benefit of mosaic augmentation is that groupings of object classes that may not exist together in the training dataset can be created (Hao, 2020). A sample image from the pedestrian dataset after mosaic augmentation has been applied is shown in Figure 8. After this method was applied, the size of the dataset for pedestrian detection grew to 312 images, and the dataset for animal detection grew to 1,324 images.

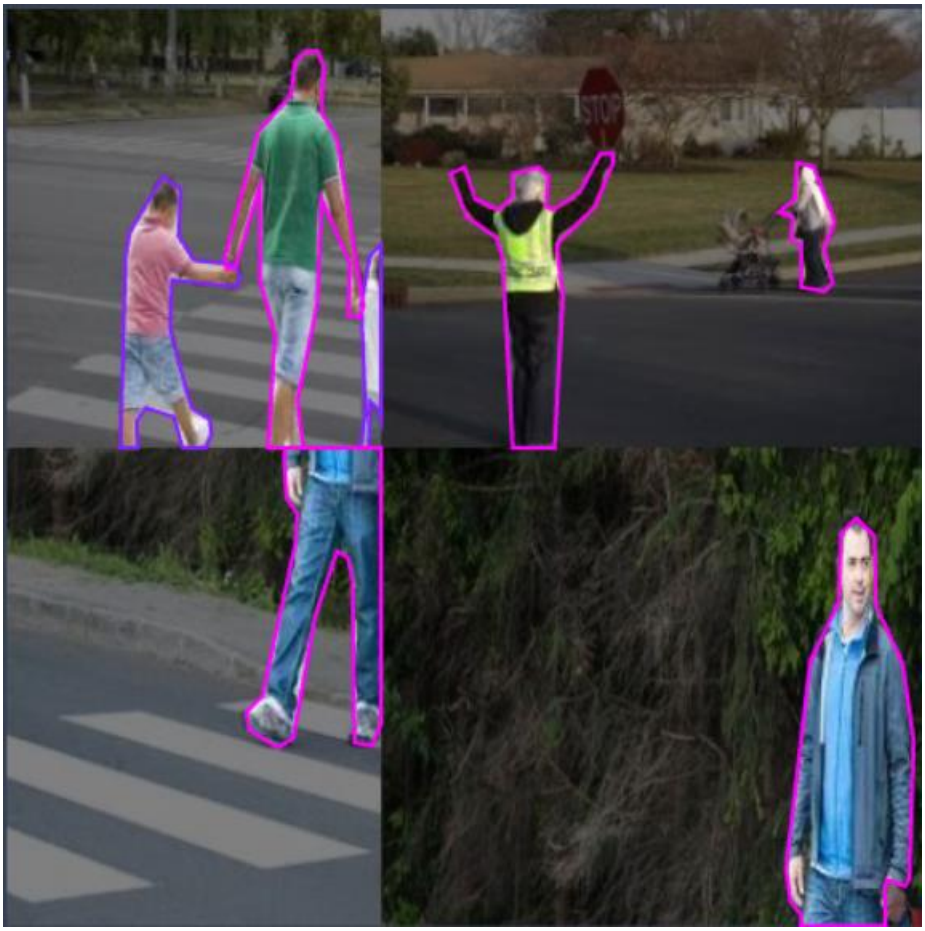

Figure 8 Sample image from the pedestrian dataset after mosaic augmentation.

3.1.5 Convolutional Neural Network

The YOLOv8 model used in this research is an example of a CNN-based approach to CV. Like other artificial neural networks (ANNs), CNNs are inspired by biological processes, specifically the neurons of the visual cortex. CNNs are primarily used for CV tasks such as image and video recognition, classification, and segmentation.

A feedforward CNN is a CNN where the connections between the nodes do not form a cycle. CNNs consist of convolutional layers, pooling layers, and fully connected layers. These layers are visualized in Figure 9.

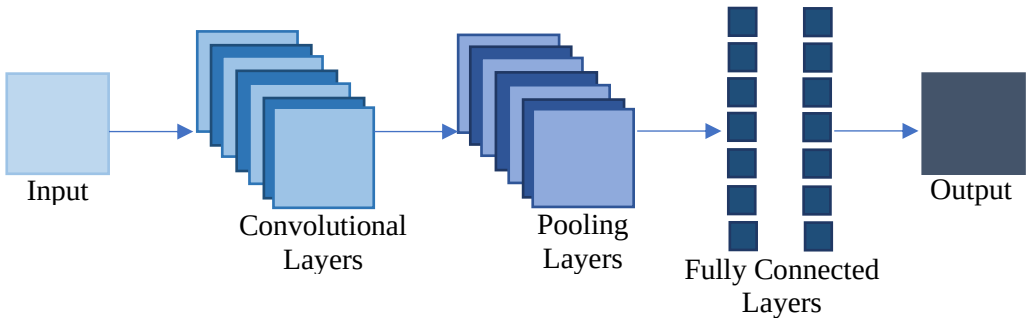

Figure 9 Convolution neural network architecture

Filters, also known as kernels, transform input data in the convolutional layers (O'Shea, 2015). Each kernel produces a unique activation map, which is passed through an activation function and combined to create the convolutional layer's final output (O'Shea, 2015). The pooling layers use a technique called max pooling or average pooling to reduce the dimensionality of the input data by taking the maximum or average value respectively of a small region of the data (O'Shea, 2015). Pooling makes the CNN less computationally expensive and easier to train (O'Shea, 2015).

Neurons form the fully connected layer, where each neuron is connected to every neuron in the previous layer (O'Shea, 2015). The flattened output of the pooling layer is the input to the fully connected layer, which also uses the features extracted by the convolutional and pooling layers, which then uses them to make a prediction (O'Shea, 2015). The fully

connected layer is processed using weights and biases to produce the final output of the CNN (O'Shea, 2015).

3.1.6 YOLOv8

At the time of this research YOLOv8 is the most current version of the YOLO OD algorithm. As this algorithm is in iterative development, the architecture of the model has undergone many changes since it was first introduced. This section will detail the general architecture of the YOLO family of OD algorithms and will discuss changes in the architecture leading up to the development of YOLOv8.

When it was first introduced in 2016, YOLO was revolutionary in its approach to OD. What was previously multiple components that enabled OD was combined into one single CNN (Redmon, 2016). YOLO used features from the entire input image to predict the bounding boxes, predicting them simultaneously across all classes (Redmon, 2016). This architecture enabled end-to-end training and real-time speed while maintaining high average precision (Redmon, 2016).

The CNN employed by the original YOLO algorithm was made up of 24 convolutional layers followed by two fully connected layers (Redmon, 2016). The alternating convolutional layers helped reduce the feature space from the previous layers (Redmon, 2016). The convolutional layers were pretrained on the ImageNet (Deng, 2009) classification task at half-resolution, before the resolution was doubled when making the final detection (Redmon, 2016).

The first improvement to the YOLO algorithm was in the form of YOLOv2 and YOLO9000, which were introduced in 2017. YOLOv2 made the following improvements on the original algorithm: batch normalization, use of a higher resolution image classifier, removal of the fully connected layers from YOLO in favor of anchor boxes, dimension clusters, direct location prediction, fine-grained features, multi-scale training, use of a new

base classification model, improved training processes, hierarchical classification, multiple dataset combination using WordTree, and joint classification and detection (Redmon, 2017). YOLO9000 was an expansion of the detection capabilities of YOLO, upping the number of object categories that could be detected to over 9000 (Redmon, 2017).

YOLOv3, proposed in 2018, makes some changes to the bounding box prediction system used in YOLO9000, in that YOLOv3 predicts an objectness score for each bounding box using logistic regression (Redmon, 2018). The YOLOv3 system predicts, in three levels of detail, where objects are in images by dividing the image into grids and predicting the location, size, and category of each object in each grid cell (Redmon, 2018). The final improvement offered by YOLOv3 over its predecessors is a new CNN for performing feature extraction that has 53 convolutional layers (Redmon, 2018).

YOLOv4, introduced in 2020, offered expansive improvements on the YOLOv3 architecture. YOLOv4 utilized a bag of freebies and bag of special approach for the backbone of the model, which included CutMix and mosaic data augmentation, DropBlock regularization, class label smoothing, mish activation, cross-stage partial connections, and multi-input weighted residual connections (Bochkovskiy, 2020). For the detector, YOLOv4 again used a bag of freebies and bag of specials approach, which included CIoU-loss, cross mini-batch normalization, DropBlock regularization, mosaic data augmentation, self-adversarial training, eliminated grid sensitivity, multiple anchors for a single ground truth, a cosine annealing scheduler, optimal hyper-parameters, random training shapes, mish activation, an SPP-block, a spatial attention model block, a path aggregation network path-aggregation block, and DIoU-NMS (Bochkovskiy, 2020).

The architecture of YOLOv5 is very similar to the architecture of YOLOv4, except YOLOv5 was the first YOLO model to break away from the Darknet backbone that formed the basis for the previous versions of the model. Instead, YOLOv5 operated fully in PyTorch,

which offered both speed and accuracy improvements (Jocher, 2020). Many components of YOLOv6 improved on YOLOv5, including network architecture, label assignment, loss function, data augmentation, industry-friendly changes, and quantization and deployment (Li, 2022).

The network structure of YOLOv6 consists of a backbone, neck, and head (Li, 2022). For large and small models respectively, the backbone is based on CSPStackRep and RepVGG blocks (Li, 2022). The neck employs a PAN topology with RepBlocks or CSPStackRep blocks for increases performance (Li, 2022). For efficiency, the head has been simplified (Li, 2022).

The label assignment approach for YOLOv6 is TAL (Li, 2022). For its loss function, YOLOv6 employs VariFocal loss for classification and SIoU/GIoU Loss for regression (Li, 2022). In terms of industry-friendly changes that were made to the algorithm, self-distillation and longer training epochs were implemented (Li, 2022). YOLOv6 is also trained with RepOptimizer to obtain PTQ-friendly weights in order to increase performance during quantization and deployment (Li, 2022).

At the same time YOLOv6 was being developed, YOLOv7 also entered the market. The authors of YOLOv7 designed several trainable bag-of-freebies methods to enable accurate real-time object detection without increasing the inference cost (Wang, 2022). They also identified and proposed methods to address issues regarding how the re-parameterized model replaces the original model and how the dynamic label assignment strategy deals with assignment to different output layers (Wang, 2022). Additionally, they proposed “extend” and “compound scaling” methods for the real-time object detector that effectively used parameters and computation, and finally, they proposed a method to effectively reduce an estimated 40% of necessary parameters and 50% of computations (Wang, 2022).

As of March 2023, YOLOv8 is the latest version of the YOLO OD algorithm. YOLOv8 expands on the features of all previous versions of the algorithm, offering improvements for the performance, flexibility, and efficiency of the algorithm. Such innovations include a new backbone network, a new anchor-free split head, and new loss functions. These improvements maintain the compact size and quick speed of the algorithm, while delivering super results.

The custom YOLOv8 implementation in this research project is designed to reduce unnecessary object detections and refine the algorithm for use in AVs, with a focus on pedestrian detection and tracking. To achieve this, we use a custom training data file. By default, YOLO models are trained on Microsoft's COCO dataset, which consists of images with 91 objects that are easily recognizable by a four-year-old (Lin, 2014).

The custom training data reduces the number of detected classes from 80 to 2, adults and children, enabling differentiation between adult and child pedestrians. Additionally, custom training data allows the object detection model to focus only on relevant objects for the task at hand. For example, an autonomous vehicle system would not need to identify a banana or carrot, both of which are included in the COCO data set.

The architecture of the YOLOv8 model used in this research consists of seven convolutional layers, eight cross stage partial (CSP) bottleneck layers with two convolutions (C2f), one spatial pyramid pooling – fast (SPPF) layer, two upsampling layers, four concatenation layers, and one segmentation layer.

The convolutional layers use a Sigmoid-weighted Linear Unit (SiLU) activation function, with two forward methods: forward and forward fuse. The forward technique applies the convolutional operation to the input tensor, followed by batch normalization and the activation function. The forward fuse method applies the activation function to the result of a transposed convolution on 2D data.

The C2f layers use two convolutions to improve the efficiency and accuracy of YOLOv8's CSPDarknet backbone. The forward pass of the C2f layers consists of multiple steps. Firstly, the input tensor is passed through a 1x1 convolution layer with double the number of hidden channels. The output is then divided into two equal chunks along the channel dimension. The first chunk is used as the primary input for further processing, while the second chunk is saved for later concatenation. The primary input is then routed through a series of bottlenecks, which use the hidden channels as both input and output channels. Finally, the outputs of the bottlenecks are concatenated with the previously stored chunk and fed through another 1x1 convolutional layer to generate the final output tensor. The C2f layers also use a forward split method to apply spatial attention to the input tensor before it is processed by the bottlenecks. The forward split works essentially the same way as the forward pass, with the exception that it applies spatial attention before concatenating the outputs with the stored chunk.

In the SPFF layer, the forward pass begins with a 1x1 convolution to cut the number of input channels in half. Then, to collect contextual information at several scales, max pooling is applied twice. The result of the first and second max pooling operations are then concatenated with the original input feature map. The concatenated feature map is then sent through another 1x1 convolution to get the final output feature map.

The upsampling layers enable efficient and accurate resizing of input data. The concatenation layers provide a convenient and efficient way to combine tensors along a specific dimension. The segmentation layer provides essential functionality for incorporating mask-based segmentation in the YOLOv8 object detection framework, enabling accurate and efficient segmentation of objects in images. The full architecture of YOLOv8 can be visualized in Figure 10.

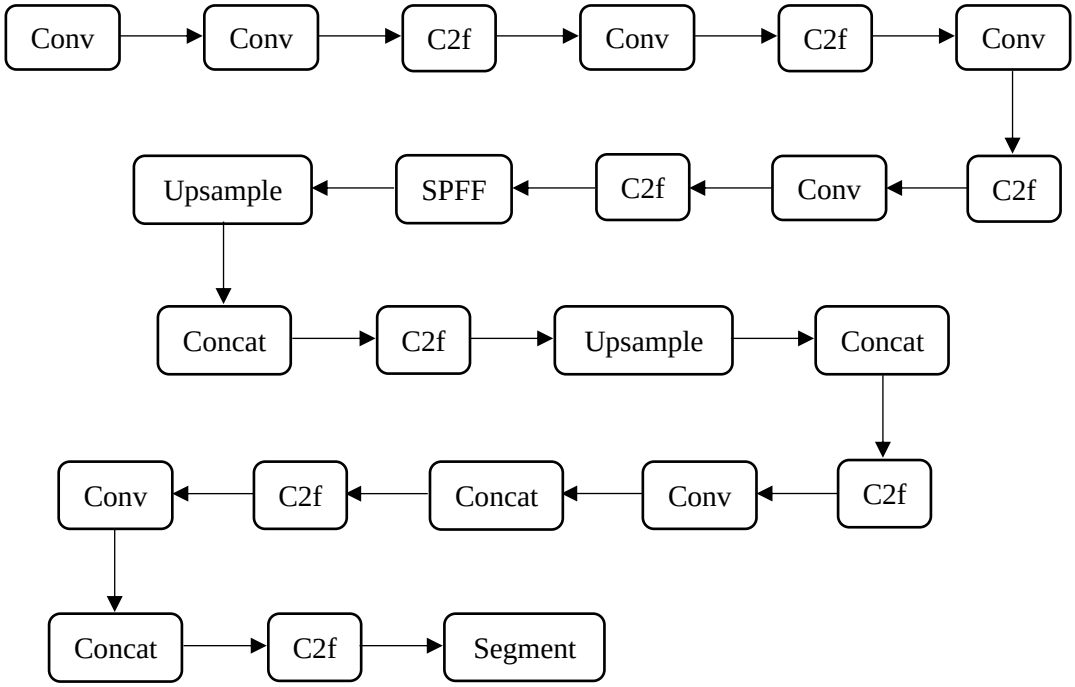

Figure 10 Architecture of the YOLOv8 model.

3.1.7 Instance Segmentation

Image segmentation is an important component of CV systems that involves partitioning image or video inputs into multiple segments and objects and has many applications, including pedestrian detection for AVs (Minaee, 2021). There are two main forms of image segmentation: semantic segmentation and instance segmentation. Semantic segmentation gives fine inference by predicting labels for every pixel in the image. Instance segmentation, on the other hand, gives different labels for separate instances of objects belonging to the same class (Hafiz, 2020).

For the purposes of this research, instance segmentation was implemented during the labeling process of the input images. Instance segmentation was chosen for several reasons, the chief reason among them being that YOLOv8 natively supports this process for training purposes. It was determined that instance segmentation should be used for the particular use cases of adult vs. child pedestrian detection as well as the animal detection because the goal of this research is to provide meaningful information to an AV about objects that cross its

path on the road. Because instance segmentation assigns different labels for every instance of an object in the scene, this can enable the proposed OD and OT system to provide information about every distinct object, so that each object can be tracked uniquely. Another consideration is that because instance segmentation detects an object based on its actual form, the detection will be more accurate than a bounding box approach because there will be less unnecessary background detected.

3.1.8 DeepSORT

This research project uses the DeepSORT algorithm for pedestrian tracking. DeepSORT is an extension of the SORT algorithm introduced in (Bewley, 2016). The architecture of DeepSORT consists of track handling and state estimation, assignment problem, matching cascade, and deep appearance descriptor (Wojke, 2017).

DeepSORT's track handling and state estimation system uses a mathematical model to predict the location of an object in subsequent frames based on its current location and movement between frames (Wojke, 2017). Considered are the object’s location on the screen, its size, and its shape (Wojke, 2017). When a new frame is processed, the model is updated using a Kalman filter, which helps to correct for any prediction errors (Wojke, 2017). Given lack of new information about the object, predictions will be based on the previous frame (Wojke, 2017).

DeepSORT extends the traditional method of solving the association between predicted Kalman states and newly arrived measurements by using the Hungarian algorithm and squared Mahalanobis distance (Wojke, 2017). The Mahalanobis distance measures the standard deviations between the detection and the mean track location, considering the uncertainty of the state estimation (Wojke, 2017). DeepSORT also uses a CNN to compute bounding box appearance descriptors (Wojke, 2017).

DeepSORT's matching cascade algorithm addresses subproblems such as uncertainty about the location of an obstructed object, which is increased by Kalman filter predictions and can lead to a spread in probability mass and a decrease in peakedness of the observation likelihood (Wojke, 2017). It also handles situations where two tracks compete for the same measurement, which can result in track fragmentation (Wojke, 2017).

The matching cascade algorithm works as follows: a set of track and detection indices as well as the maximum age are given as input, the association cost matrix and admissible associations matrix are calculated, and a linear assignment problem for tracks of increasing age is solved (Wojke, 2017). Tracks that have not been associated with a detection past a certain point are then selected and the linear assignment between these tracks and unmatched detections is solved (Wojke, 2017). Finally, it updates the set of matches and unmatched detections and runs intersection over union (IoU) association on unconfirmed and unmatched tracks past a certain age to account for sudden appearance changes (Wojke, 2017).

The final part of the DeepSORT algorithm is the deep appearance descriptor, which uses a pre-trained CNN for deep metric learning in a people tracking context (Wojke, 2017). This descriptor is trained offline, before the online tracking application, and employs simple nearest neighbor queries without further metric learning (Wojke, 2017).

3.1.9 Speed Estimation

The algorithm for speed estimation can be optimized by only using calculations from a subset of the video. Two areas are designated in the video, so that when an object enters the first area, the timestamp of the video at that point, *A_E*, and the object's position, *A_P*, are recorded. Similarly, the timestamp at which the object enters the second area, *B_E*, and its position, *B_P*, are recorded. To estimate the object's speed, the elapsed time, *T*, from its entry to its exit must be calculated.

$$T = B\_E - A\_E \tag{1}$$

One of the biggest challenges when attempting to estimate the speed of a tracked object is measuring the actual time that passes between the object entering and exiting an area. If done improperly, the estimated speed will not be accurate, and will be tied to the processing speed of the computer. The method proposed in this research aims to prevent this by taking the processing speed of the frames themselves into the calculation. At the start of processing, the timestamp is logged, *S*, as well as at the end of processing, *E*. A running total for the processing time, *PT*, is kept, and both timestamps for an object's entry, *N*, and exit, *X*, into an area are logged. The difference between these two variables is incremented to the running total.

$$PT += (X - N) \tag{2}$$

At the end of processing, the total time, *TT*, taken is calculated:

$$TT = (E - S) - PT \tag{3}$$

As well as the distance, *D*, the object has traveled:

$$D = B\_P \ - A\_P \tag{4}$$

The average feet or meters traveled per second, *V*, must be calculated in order to estimate the speed. This calculation uses both *T* and *D*. The formula works for both the Imperial system and the metric system. Both units of measurement are considered.

$$V = \frac{D}{T} \tag{5}$$

If the estimated speed, *ES*, is measured in miles per hours, it is converted from *V* using the constant value of 1.467:

$$ES = \frac{V}{1.467} \tag{6}$$

If *ES* is measured in kilometers per hour, it is converted from *V* using the constant value of 3.6:

$$ES = V \times 3.6 \tag{7}$$

3.1.10 Trajectory Prediction

To predict the trajectory of pedestrians, historical position data from the DeepSORT implementation is logged. This data is then split into training and testing sets using an 80-20 split. The training set is used to fit a k-Nearest Neighbors (k-NN) regression model, while the test set is used to evaluate the accuracy of the model's predictions.

The k-NN regressor algorithm is a type of supervised learning algorithm used for regression tasks, where the goal is to predict a continuous numeric value, rather than a class label. The k-NN regressor algorithm is based on the principle that similar data points tend to have similar output values. The working process for the k-NN regressor is described in the following paragraph.

Based on the position data exported from DeepSORT, a value for *k* is chosen. For the purposes of this research, a *k* value of two was used. After the value of k was selected, the distance between a given data point and all of the other points in the training set was calculated based on Euclidean distance. Once the distances have been calculated, the nearest neighbors are determined. Once the neighbors have been identified, the average value of the output variable for the k nearest neighbors is calculated, and becomes the predicted output value for the new data point. This process is repeated until all predictions are complete.

3.1.11 Performance Evaluation

It is important to evaluate the classification process and measure the performance of the custom YOLOv8 implementation. YOLOv8 has several built-in metrics to evaluate its performance, involving the use of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) values. These metrics are defined as:

1) *Precision (P)*: The percentage of correctly classified positive cases relative to the cases classified as positive.

$$P = \frac{TP}{TP + FP} \tag{8}$$

2) *Recall (R)*: The percentage of positive cases that were successfully classified as positive.

$$R = \frac{TP}{TP + FN} \tag{9}$$

3) *mean Average Precision (mAP)*: The average precision (AP) of each class over a number of classes (n).

$$mAP = \frac{1}{n}\sum_{i=1}^{N} AP_i \tag{10}$$

To evaluate the performance of the k-NN regressor at accurately predicting trajectory, the following metrics are used:

1) *Mean Squared Error (MSE)*: The average squared difference between the estimated values (Y) and the actual values ($\hat{Y}$).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \tag{11}$$

2) *Coefficient of Determination (R^2) Score*: The proportion of variation in the dependent variable that is predictable from the independent variables, which is measured by the residual sum of squares (SS_{res}) and the total sum of squares (SS_{tot}).

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{12}$$

CHAPTER IV: EXPERIMENTAL RESULTS

The first part of this chapter discusses the experimental results of the proposed adult vs. child pedestrian detection and tracking system. The second part of this chapter discusses the experimental results of the proposed methodology for the proposed animal detection and tracking system.

4.1 Experimental Results for the Pedestrian Detection and Tracking System

Because of the wide number of YOLO models that are available to the public, the proposed adult and child pedestrian detection system was tested on several of the most popular models: YOLOv3, YOLOv4, YOLOv5, YOLOv6, YOLOv7, and YOLOv8. The results of these experiments are shown in Table 1. The experiments show that the YOLOv8 model performs the best compared to the previous YOLO models for the task of adult and child pedestrian detection when the detection was being performed using the bounding box method.

The P of YOLOv8 is given as 79.3% overall, with adults having 85.1% P and children having 73.6%. The overall R is 79.7%, representing 74% for adults and 85.4% for children. The mAP for an intersection-over-union (IoU) threshold of 50% (mAP50) is 83.5% overall, 84.5% for adults, and 82.5% for children. When the IoU threshold is changed to measure all thresholds from 50% to 95% at an interval of 5% (mAP50-95), the mAP becomes 49.9% overall, 50.8% for adults, and 48.9% for children.

Once it was determined that of the tested YOLO models, YOLOv8 performed the best, instance segmentation was introduced into the model, changing the results slightly. With segmentation added, the overall P was 78.7%, 67.8% for adults, and 89.5% for children. The overall R became 78.6, with the R for adults becoming 96.7%, and 60.4% for children. The mAP50 score changed to 82.2% overall, 91.2% for adults, and 73.3% for children. Finally, the mAP50-95 score changed to 71.9% overall, 79.5% for adults,

and 64.3% for children. From these observations, it is noted that adding segmentation to the model improved the P for children, the R and mAP50 for adults, and in every category for mAP50-95. However, segmentation reduced the values for the other categories. Because the number of improved categories is evenly split between base YOLOv8 and YOLOv8 with segmentation, and because the added merits of implementing instance segmentation have already been discussed in this thesis, for the further tests of the algorithm, YOLOv8 with instance segmentation was used.

Table 1 Comparison of YOLO models for adult and child pedestrian detection.

Class	Images	Instances	P	R	mAP50	mAP50-95
YOLOv5						
Overall	45	140	83.1%	72%	78.7%	45.7%
Adult	45	92	84.3%	66.3%	76.9%	44.8%
Child	45	48	82%	77.8%	80.5%	46.6%
YOLOv7						
Overall	45	140	15.7%	30%	14.3%	3.42%
Adult	45	92	21.3%	39.1%	22.4%	5.1%
Child	45	48	10.1%	20.8%	6.3%	1.73%
YOLOv8						
Overall	45	140	**79.3%**	**79.7%**	**83.5%**	49.9%
Adult	45	92	**85.1%**	74%	84.5%	50.8%
Child	45	48	73.6%	**85.4%**	**82.5%**	48.9%
YOLOv8 + Segmentation						
Overall	45	140	78.7%	78.6%	82.2%	**71.9%**
Adult	45	92	67.8%	**96.7%**	**91.2%**	**79.5%**
Child	45	48	**89.5%**	60.4%	73.3%	**64.3%**

Once the appropriate model that was to be used for the adult and child pedestrian detection system was determined, tests were conducted to determine the effectiveness of augmenting the training data of the model. The results of these tests are reported in Table 2. The data augmentation techniques that were studied are as follows: horizontal flipping, 90^{T} rotation clockwise and counterclockwise, random cropping by 20%, random rotation by 15°, horizontal and vertical random shearing at 15°, grayscale conversion of 25%, hue changes of 25°, saturation changes of 25%, brightening and darkening by 25%, exposure changes of 25%, 10px blurring, noisiness of 5%, three 10% cutouts, and mosaic augmentation.

The experiments show that adding 5% of noise to the training data offers the highest R for adults, 95.7%, while flipping horizontally and rotating clockwise and counterclockwise by 90° both give the highest R for children, 95.8%. In terms of P, rotating clockwise and counterclockwise by 90° offers the highest P across all categories, with 95.6% overall, 97.7% for adults, and 93.6% for children. However, in terms of mAP values, the mosaic data augmentation offers the highest values for mAP50 and mAP50-95 across all categories. The mAP50 values are 97.2% overall, 97.9% for adults, and 96.6% for children. The mAP50-95 values, comparatively, are 89.6% overall, 91.9% for adults, and 87.3% for children. Because the mosaic data augmentation offers the most improvements overall to the original YOLOv8 and instance segmentation model, all future experiments were conducted with this augmentation added.

Table 2 Comparison of data augmentation techniques for adult and child pedestrian detection.

Class	Images	Instances	P	R	mAP50	mAP50-95
Saturation						
Overall	45	140	83.8%	75.6%	84.4%	71.4%
Adult	45	92	88.8%	80.4%	86%	73.9%

Child	45	48	78.8%	70.8%.	82.9%	68.9%
Noise						
Overall	45	140	82.1%	79.6%	85.4%	73%
Adult	45	92	75.8%	**95.7%**	92.3%	80.4%
Child	45	48	88.4%	63.6%	78.5%	65.6%
Brightness						
Overall	45	140	79.7%	81.4%	85.7%	74.1%
Adult	45	92	82.2%	85.3%	90.8%	80.9%
Child	45	48	77.2%	77.6%	80.6%	67.4%
Shear						
Overall	45	140	82.5%	78.4%	87.4%	69.7%
Adult	45	92	80%	90.2%	92%	74.7%
Child	45	48	85%	66.7%	82.7%	64.7%
Grayscale						
Overall	45	140	83.7%	81.1%	87.4%	72.8%
Adult	45	92	89.9%	87.3%	91.8%	77.2%
Child	45	48	77.5%	75%	83%	68.4%
Cutout						
Overall	45	140	82.5%	82.8%	87.9%	75.7%
Adult	45	92	86%	86.5%	91.1%	79.1%
Child	45	48	79%	79.2%	84.7%	72.4%
Exposure						
Overall	45	140	85.2%	82.7%	88%	73.6%
Adult	45	92	87.7%	85.6%	91.6%	78.2%

Child	45	48	82.7%	79.7%	84.4%	69%
Hue						
Overall	45	140	78.6%	86.2%	88.9%	74.6%
Adult	45	92	77.3%	89.1%	92.1%	78.7%
Child	45	48	80%	83.3%	85.7%	70.5%
Crop						
Overall	45	140	82.9%	87%	88.9%	75.1%
Adult	45	92	79.2%	93.5%	91.1%	80.6%
Child	45	48	86.6%	80.6%	86.7%	69.7%
Blur						
Overall	45	140	89.6%	78.3%	89.5%	75.3%
Adult	45	92	89.9%	87.2%	92.2%	79.9%
Child	45	48	89.3%	69.4%	86.7%	70.8%
Random Rotate						
Overall	45	140	81.7%	89.7%	89.5%	75.3%
Adult	45	92	87.5%	88%	91.6%	79.2%
Child	45	48	75.8%	91.3%	87.4%	71.5%
Flip						
Overall	45	140	91.9%	94.1%	95.2%	88.3%
Adult	45	92	95.5%	92.4%	97%	90.9%
Child	45	48	88.3%	**95.8%**	93.3%	85.7%
Rotate by 90°						
Overall	45	140	**95.6%**	94%	96.3%	87.2%
Adult	45	92	**97.7%**	92.2%	97.2%	90.4%

Child	45	48	**93.6%**	**95.8%**	95.5%	83.9%
Mosaic						
Overall	45	140	93.3%	91.9%	**97.2%**	**89.6%**
Adult	45	92	93.5%	94.3%	**97.9%**	**91.9%**
Child	45	48	93%	89.6%	**96.6%**	**87.3%**

The YOLO algorithm has several parameters that can affect its performance. Image size, batch size, the number of epochs, and the weights used are examples of these changeable parameters. Several experiments were conducted to determine which parameter configuration offered the best performance. Table 3 shows the results of the custom object detection method using the best configuration settings of an image size of 1280, batch size of 4, 300 epochs, and YOLOv8x6 weights.

The final results of the proposed adult and child pedestrian detection system are described. The overall P is 95.8% with the adult class having a P of 96.5% and the children class having a P of 95.1%. The overall R is 93.4%, with the adult class having a R of 88.9% and the children class having an R of 97.9%. The mAP50 values are 98.4% overall, 97.9% for adults, and 98.9% for children. The mAP50-95 values are 89.1% overall, 91.5% for adults, and 86.7% for children.

Table 3 Final results of the adult and child pedestrian detection system.

Class	Images	Instances	P	R	mAP50	mAP50-95
Overall	45	140	95.8%	93.4%	98.4%	89.1%
Adult	45	92	96.5%	88.9%	97.9%	91.5%
Child	45	48	95.1%	97.9%	98.9%	86.7%

Since this proposed adult and child pedestrian detection system is designed to be implemented for use in AVs, it is critical that the algorithm operates in real-time. Table 4 shows an analysis of different speed metrics for the algorithm. The preprocessing speed of the algorithm is 4.3 ms, the inference speed is 28.3 ms, there is a loss of 0.0 ms, and the postprocessing per image takes 1.1 ms. These values show that the proposed adult and child pedestrian detection system is capable of operating in real-time, which signals that it can be implemented into AVs effectively.

Table 4 Analysis of the speed of the proposed adult and child pedestrian detection system.

Speed			
Preprocessing	**Inference**	**Loss**	**Postprocessing per Image**
4.3 ms	28.3 ms	0.0 ms	1.1 ms

In addition to this raw data that is generated by YOLOv8, there are many other metrics that can be used to evaluate the performance of the algorithm. Figure 11 shows a sample of six images after the algorithm has been run, showing the predicted class for each detected object. Table 5 shows the confusion matrix for the system. Figure 12 shows the F1-confidence curve for the adult and child pedestrian detection system. Figure 13 shows the PR curve for the proposed detection system. Figure 14 shows the P-confidence curve for the system. Figure 15 shows the R-confidence curve. Figure 16 shows several visualizations of the data labels in the proposed system. Figure 17 shows the correlogram for the data labels in the proposed system. Figure 18 shows several performance metrics for the algorithm given by scatter plots.

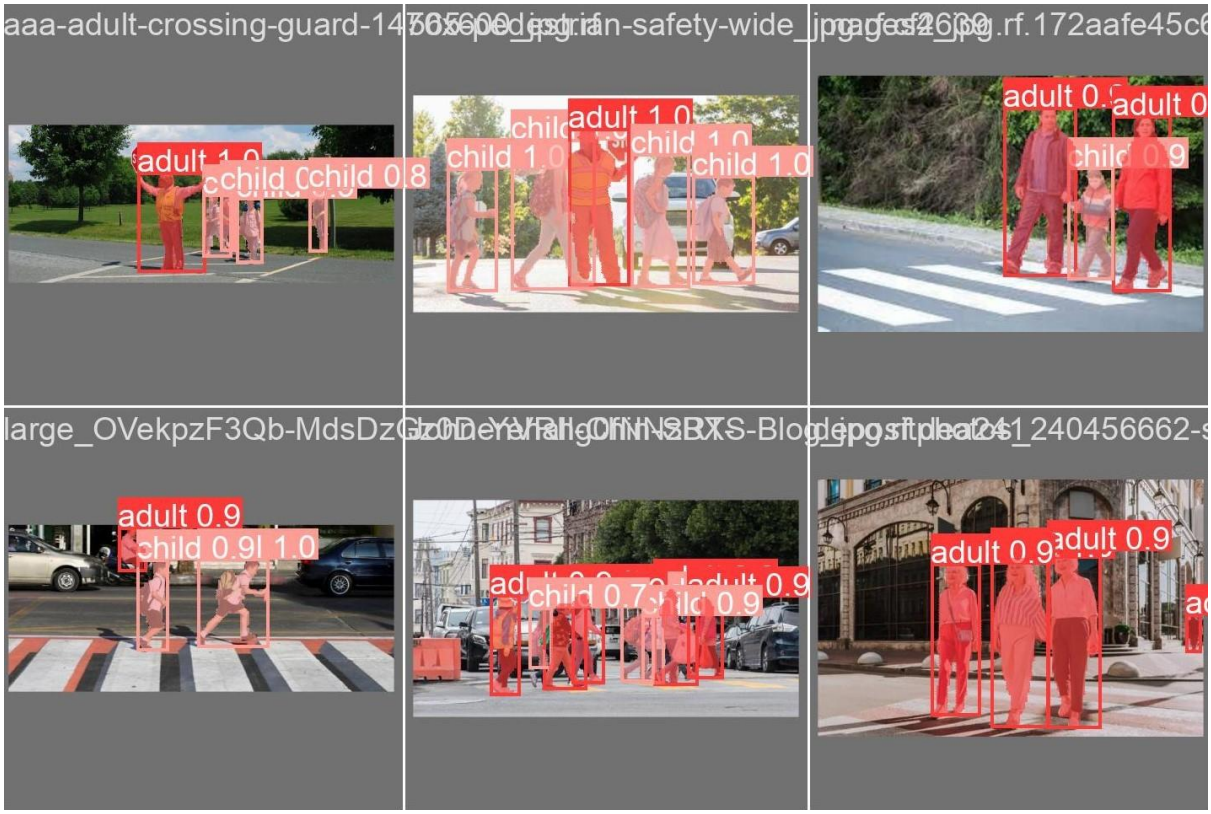

Figure 11 A sample of the results of the adult and child pedestrian detection system.

Table 5 Confusion matrix for the adult and child pedestrian detection system.

		Actual	
		adult	child
Predicted	adult	90	2
	child	1	48

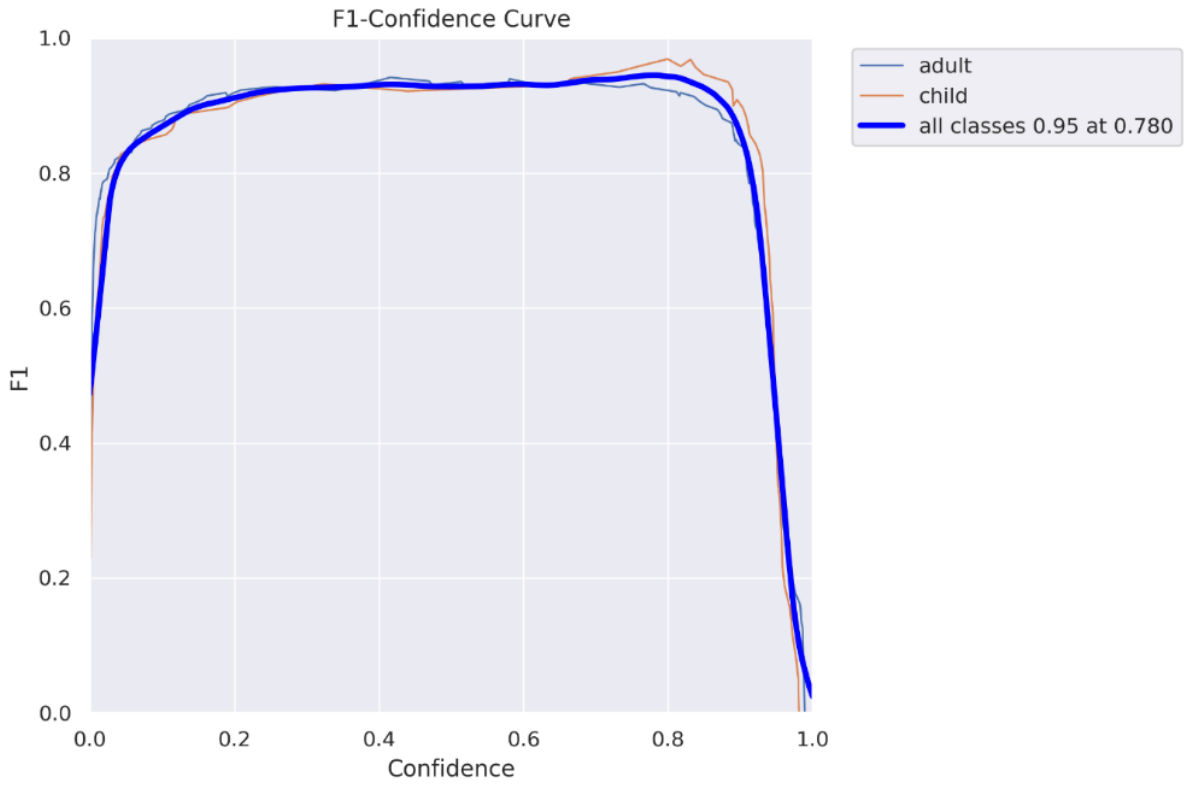

Figure 12 The F1-confidence curve for the adult and child pedestrian detection system.

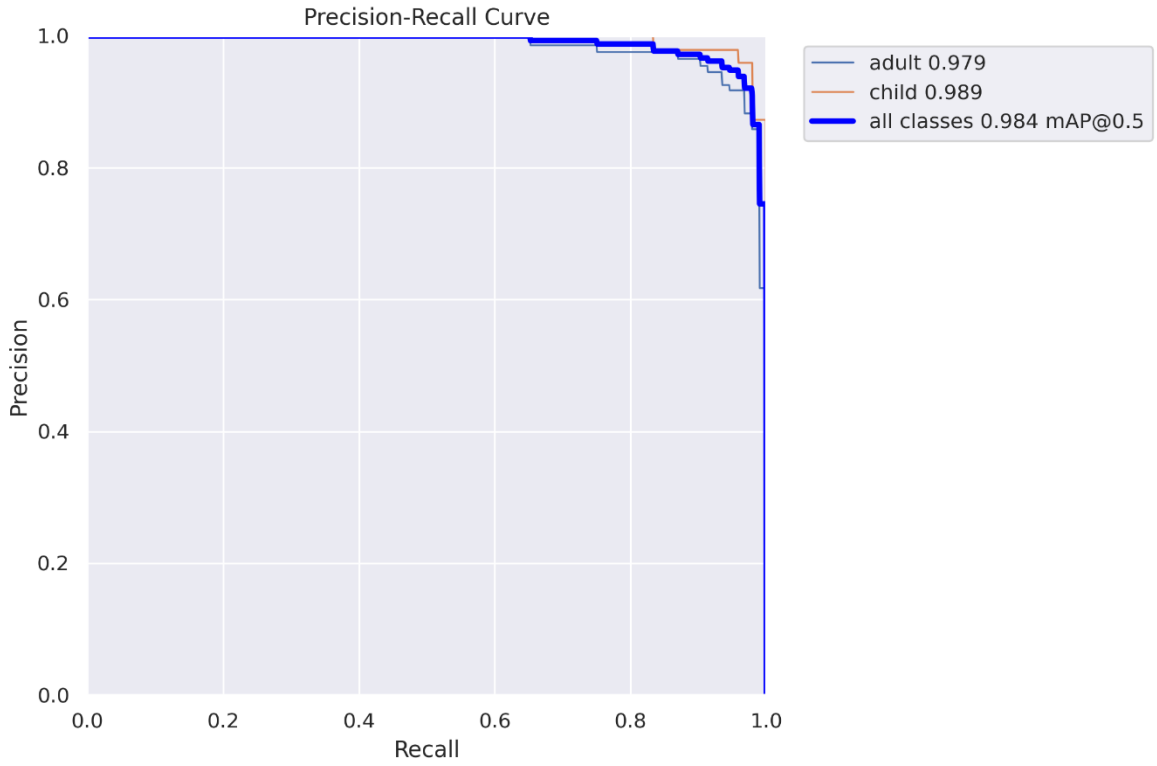

Figure 13 The precision-recall curve for the adult and child pedestrian detection system.

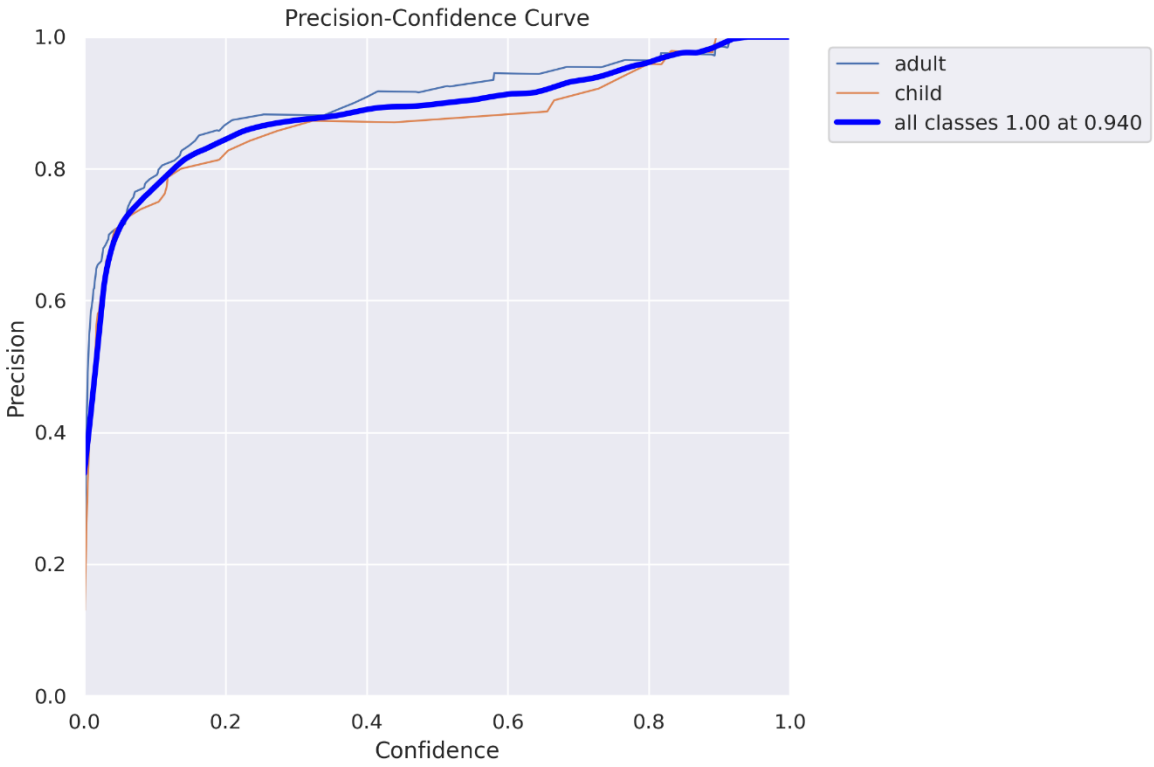

Figure 14 The precision-confidence curve for the adult and child pedestrian detection system.

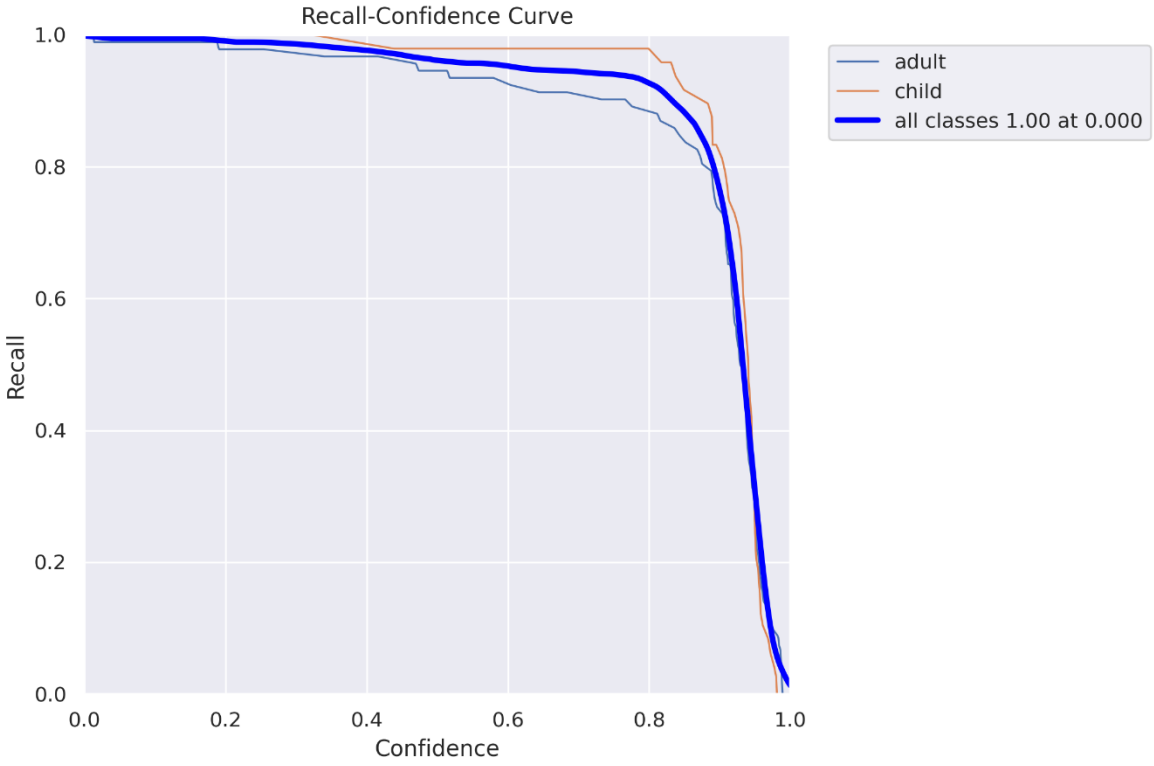

Figure 15 The recall-confidence curve for the adult and child pedestrian detection system.

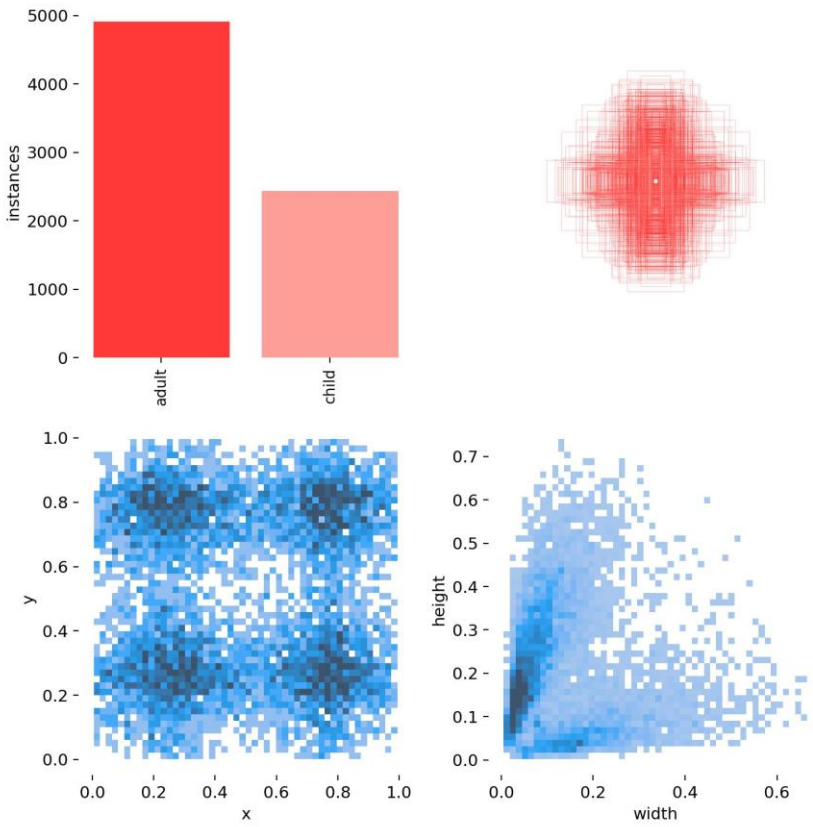

Figure 16 Visualizations of the data labels in the adult and child pedestrian detection system.

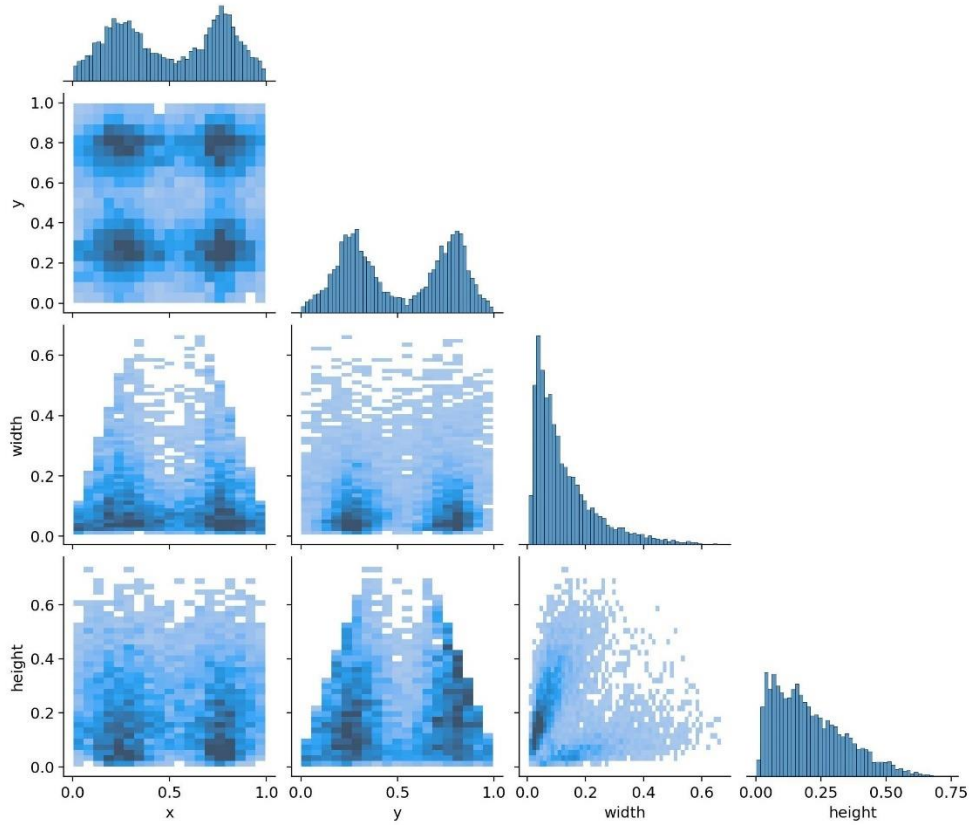

Figure 17 The correlogram for the data labels in the adult and child pedestrian detection system.

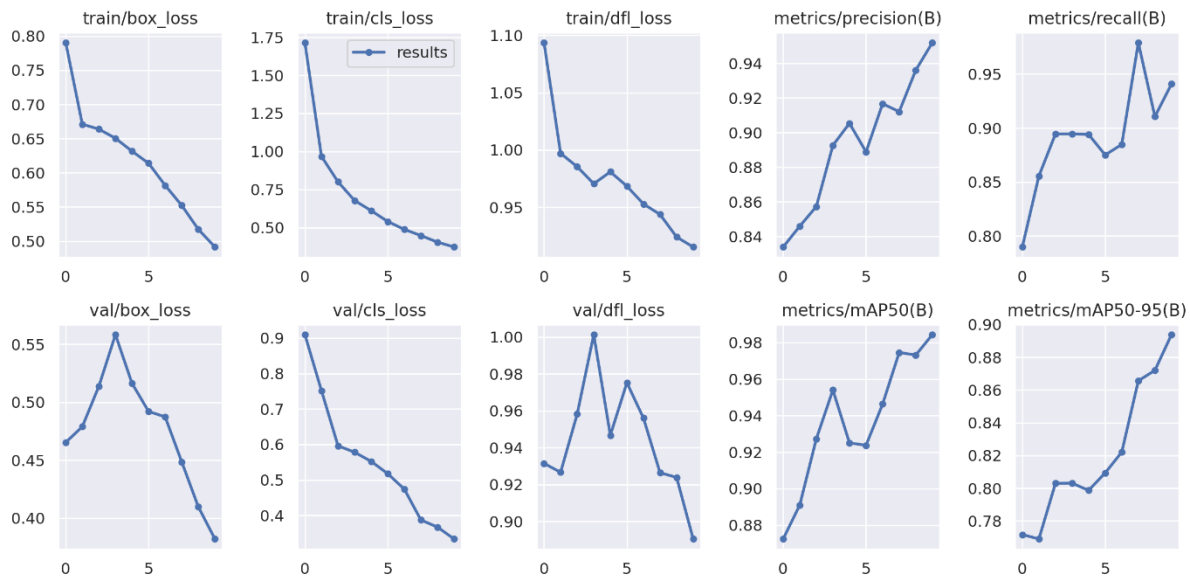

Figure 18 Scatter plots of various measures of performance for the adult and child pedestrian detection system.

Once the custom YOLOv8 model has been trained, the generated weights file is then fed into the DeepSORT OT model. An example of the combined YOLOv8 + DeepSORT model for adult and child pedestrian tracking is shown in Figure 19.

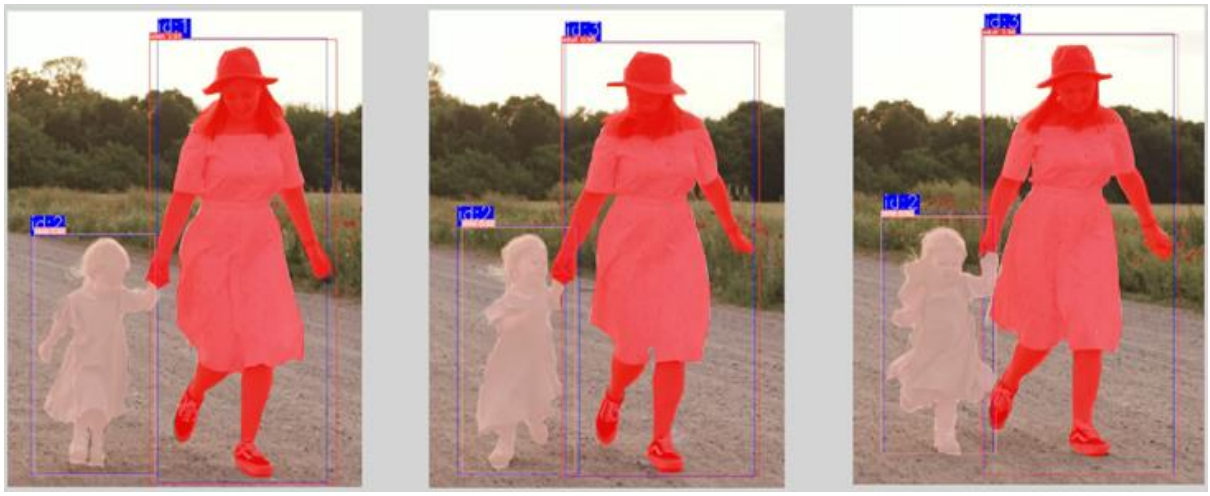

Figure 19 Sample results of the adult and child pedestrian tracking system.

4.2 Experimental Results for the Animal Detection and Tracking System

Table 6 shows the results of the YOLOv8 segmentation model with the mosaic augmentation applied. The training settings for the algorithm are the same as for the adult and child pedestrian detection and tracking system. Figure 20 shows a sample of sixteen images after the algorithm has been run, showing the predicted class for each detected object. Figure

21 shows the confusion matrix for the system. Figure 22 shows the F1-confidence curve for the animal detection system. Figure 23 shows the PR curve for the proposed detection system. Figure 24 shows the P-confidence curve for the system. Figure 25 shows the R-confidence curve. Figure 26 shows several visualizations of the data labels in the proposed system. Figure 27 shows the correlogram for the data labels in the proposed system. Figure 28 shows several performance metrics for the algorithm given by scatter plots.

Table 6 Results of the proposed animal detection system.

Class	Images	Instances	P	R	mAP50	mAP50-95
Overall	103	104	86.5%	97%	99.5%	93.2%
Alligator	103	2	98.4%	100%	99.5%	89.5%
Armadillo	103	2	83.3%	100%	99.5%	99.5%
Badger	103	2	100%	95.9%	99.5%	99.5%
Barn Owl	103	2	82.2%	100%	99.5%	99.5%
Bat	103	2	82.7%	100%	99.5%	99.5%
Bison	103	2	82.6%	100%	99.5%	92.2%
Bull Snake	103	2	83.4%	100%	99.5%	99.5%
Chimney Swift	103	2	81.8%	100%	99.5%	99.5%
Chipmunk	103	2	84.2%	100%	99.5%	99.5%
Cow	103	2	82.6%	100%	99.5%	99.5%
Crow	103	2	86.9%	100%	99.5%	94.6%
Deer	103	2	82.4%	100%	99.5%	99.5%
Desert Tortoise	103	2	79.5%	100%	99.5%	99.5%
Elk	103	2	100%	90.8%	99.5%	90.1%
Garter Snake	103	2	85.6%	100%	99.5%	94.6%

Gopher Tortoise	103	2	83.1%	100%	99.5%	99.5%
Green Snake	103	2	82.5%	100%	99.5%	79.7%
Groundhog	103	2	85%	100%	99.5%	92.1%
Herring Gull	103	2	78.1%	100%	99.5%	99.5%
Hognose Snake	103	2	100%	96.5%	99.5%	70.6%
House Sparrow	103	2	89%	100%	99.5%	60.7%
Jackrabbit	103	2	77.2%	100%	99.5%	99.5%
Marmot	103	2	70.6%	100%	99.5%	99.5%
Meadowlark	103	2	83.6%	100%	99.5%	94.5%
Moose	103	2	89.4%	100%	99.5%	75%
Mouse	103	2	100%	77.2%	99.5%	94.6%
Muskrat	103	2	73.3%	100%	99.5%	94.5%
Northern Oriole	103	2	84.3%	100%	99.5%	64.8%
Opossum	103	2	97%	100%	99.5%	99.5%
Painted Turtle	103	2	100%	54.8%	99.5%	99.5%
Pigeon	103	2	81.8%	100%	99.5%	94.6%
Porcupine	103	2	86.7%	100%	99.5%	79.9%
Rabbit	103	2	81.1%	100%	99.5%	94.5%
Raccoon	103	2	86.9%	100%	99.5%	88.7%
Rat	103	2	82.6%	100%	99.5%	99.5%
Rattlesnake	103	2	88%	100%	99.5%	75.5%
Red-headed Woodpecker	103	2	93.9%	100%	99.5%	99.5%
Red-winged Blackbird	103	2	81.2%	100%	99.5%	99.5%

Ring-necked Pheasant	103	2	85.6%	100%	99.5%	89.8%
Ring-necked Snake	103	2	82.8%	100%	99.5%	94.6%
Roadrunner	103	2	79.9%	100%	99.5%	99.5%
Robin	103	2	84.2%	100%	99.5%	99.5%
Skunk	103	2	100%	69.2%	99.5%	94.7%
Snapping Turtle	103	2	100%	96.9%	99.5%	99.5%
Snowshoe Hare	103	2	100%	90.7%	99.5%	82.2%
Softshell Turtle	103	2	82.9%	100%	99.5%	99.5%
Squirrel	103	2	100%	76.6%	99.5%	89.5%
Starling	103	2	81.7%	100%	99.5%	94.6%
Toad	103	2	87.3%	100%	99.5%	99.5%
Vole	103	2	74.9%	100%	99.5%	99.5%
Yellow-shafted Flicker	103	2	83.6%	100%	99.5%	99.5%

Figure 20 A sample of the results of the animal detection system.

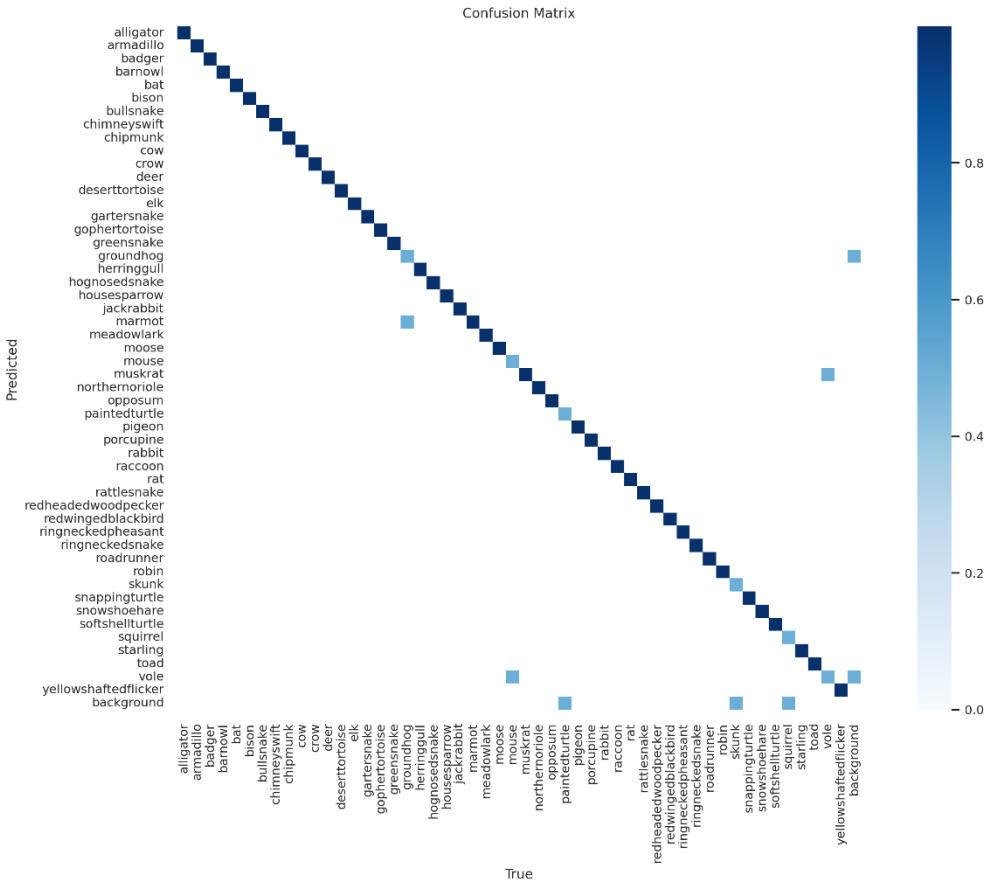

Figure 21 The confusion matrix of the proposed animal detection system.

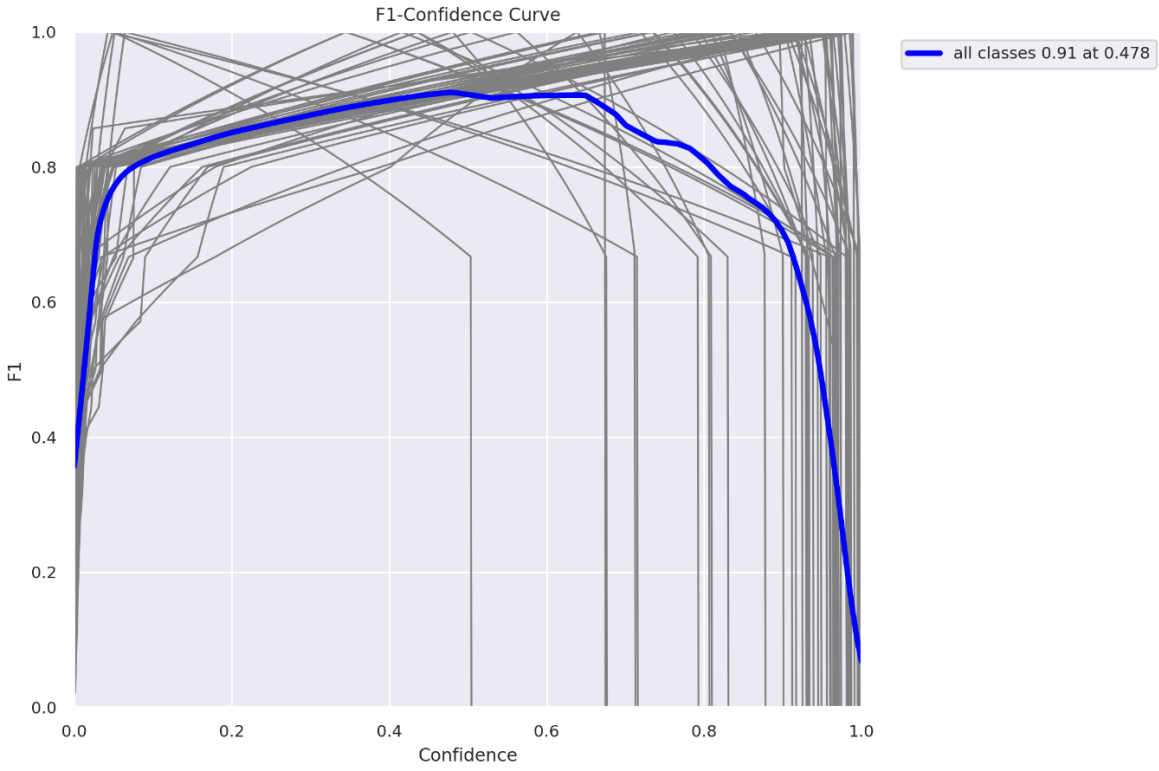

Figure 22 The F1-confidence curve for the proposed animal detection system.

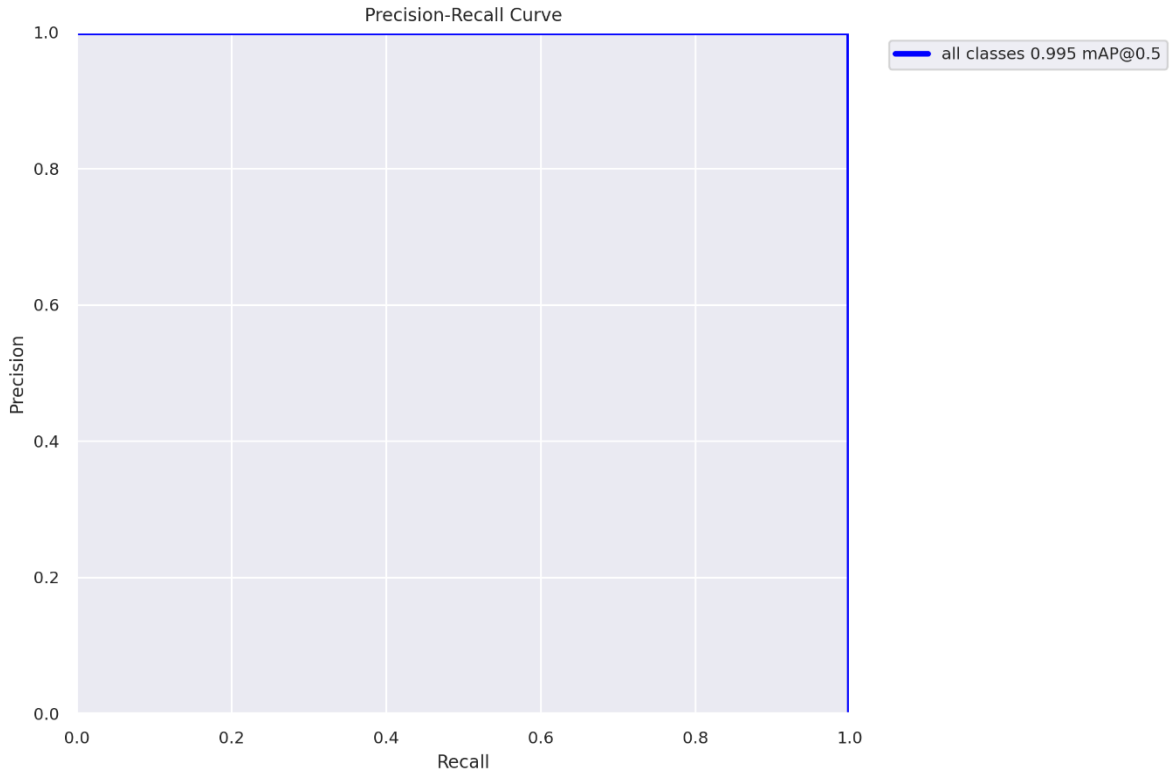

Figure 23 The PR curve for the proposed animal detection system.

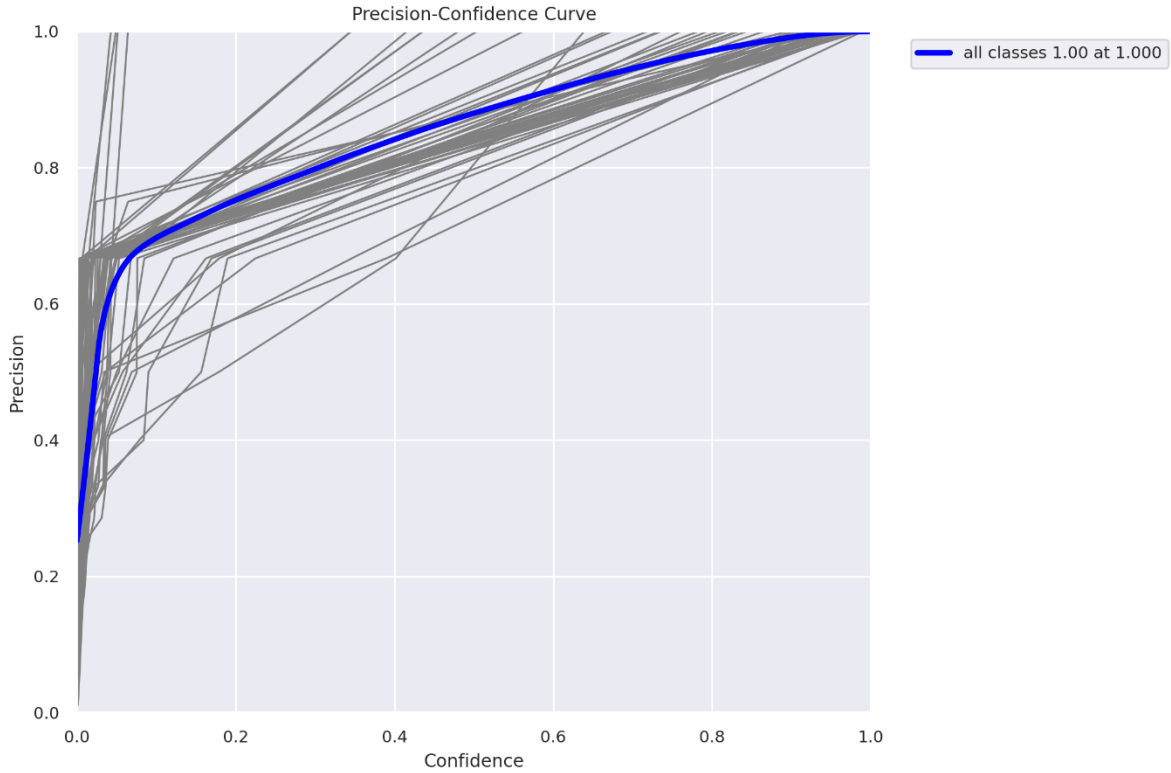

Figure 24 The P-confidence curve for the proposed animal detection system.

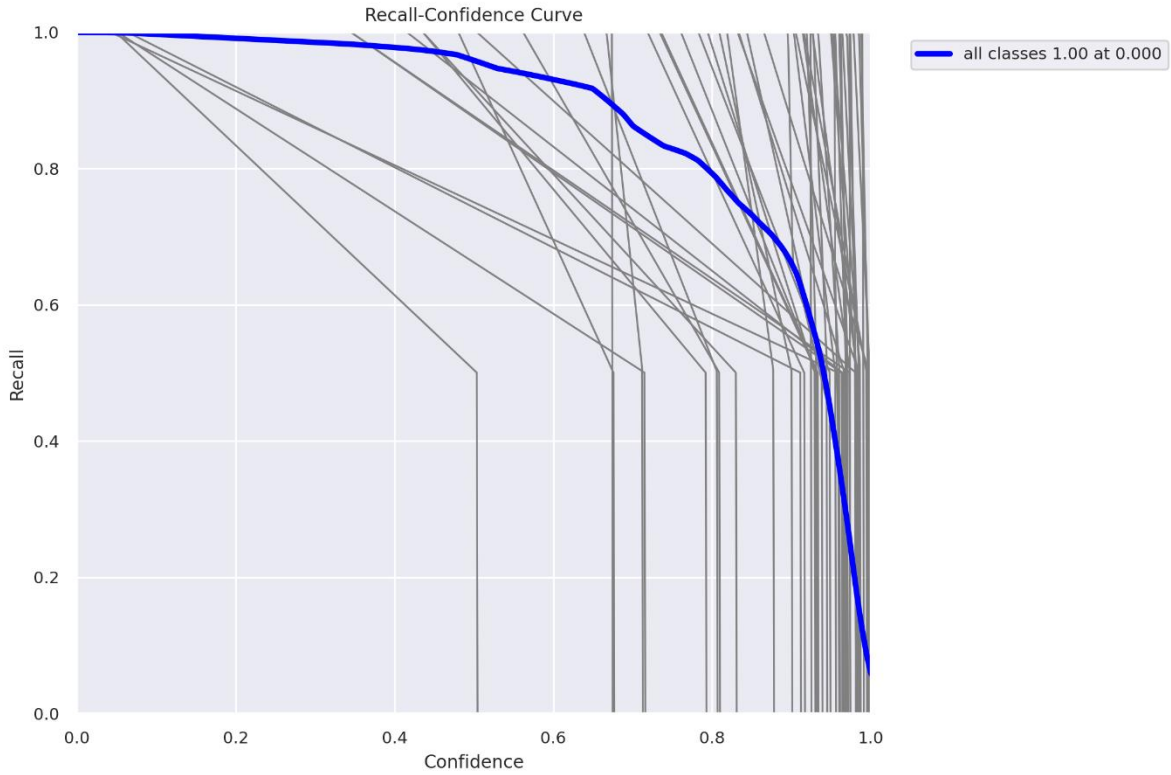

Figure 25 The R-confidence curve for the proposed animal detection system.

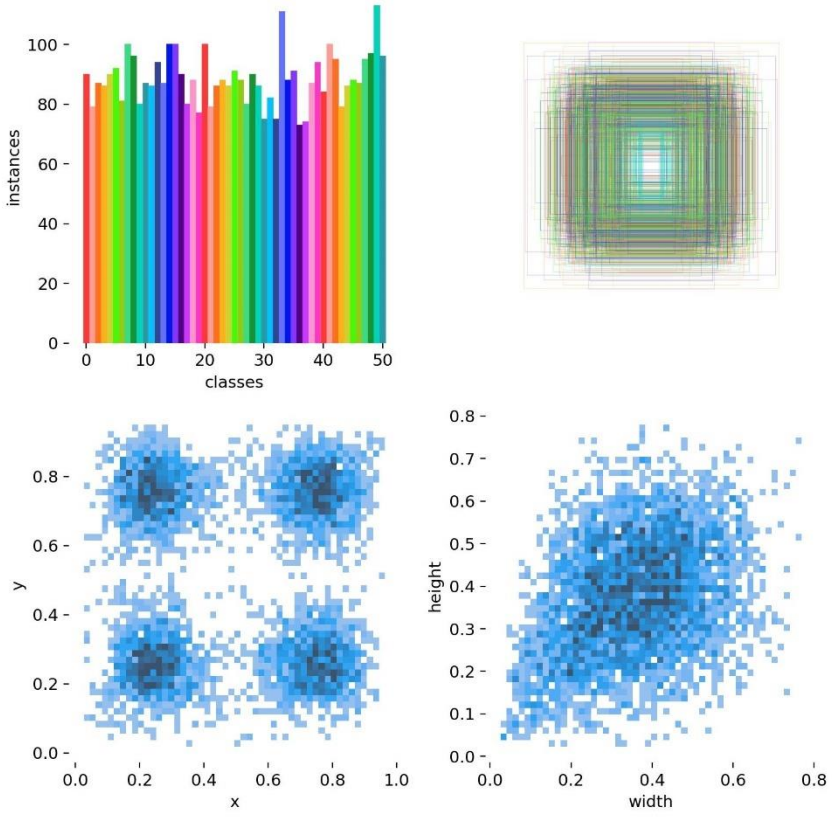

Figure 26 Visualizations of the data labels in the proposed animal detection system.

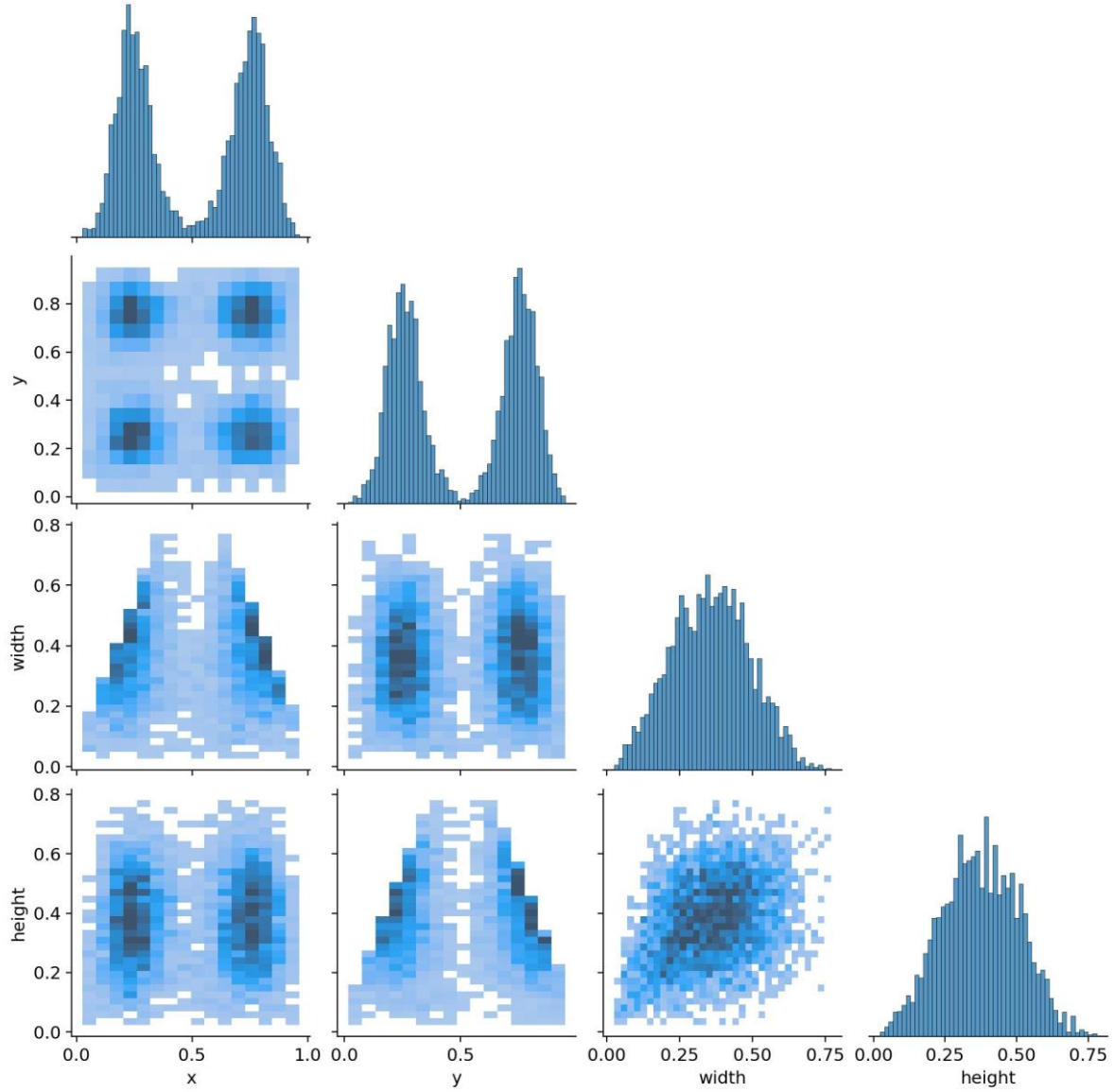

Figure 27 The correlogram for the data labels in the proposed animal detection system.

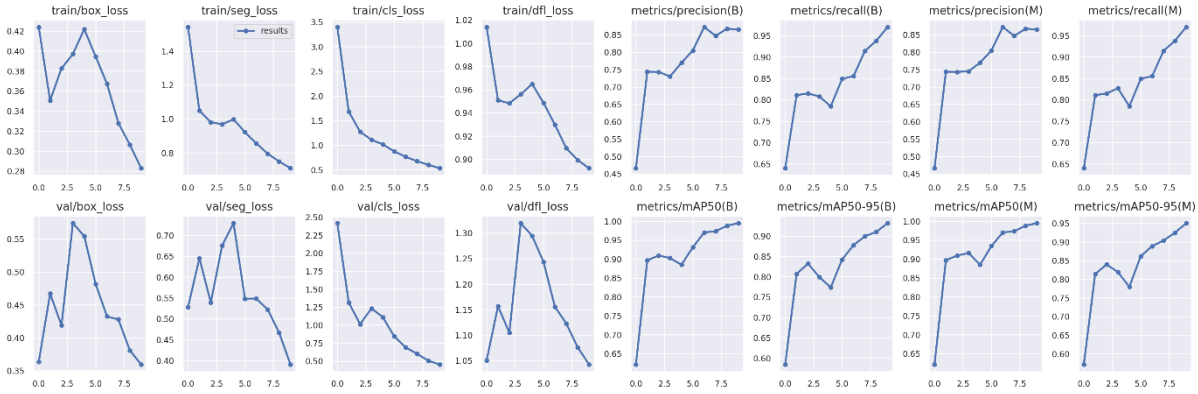

Figure 28 Scatter plots of various measures of performance for the proposed animal detection system.

As with the tracking system for adult and child pedestrians, once the OD model has been trained with the animal data, the generated weights are fed into the DeepSORT model. A sample result of an animal from the dataset, a bison, being tracked is shown in Figure 29.

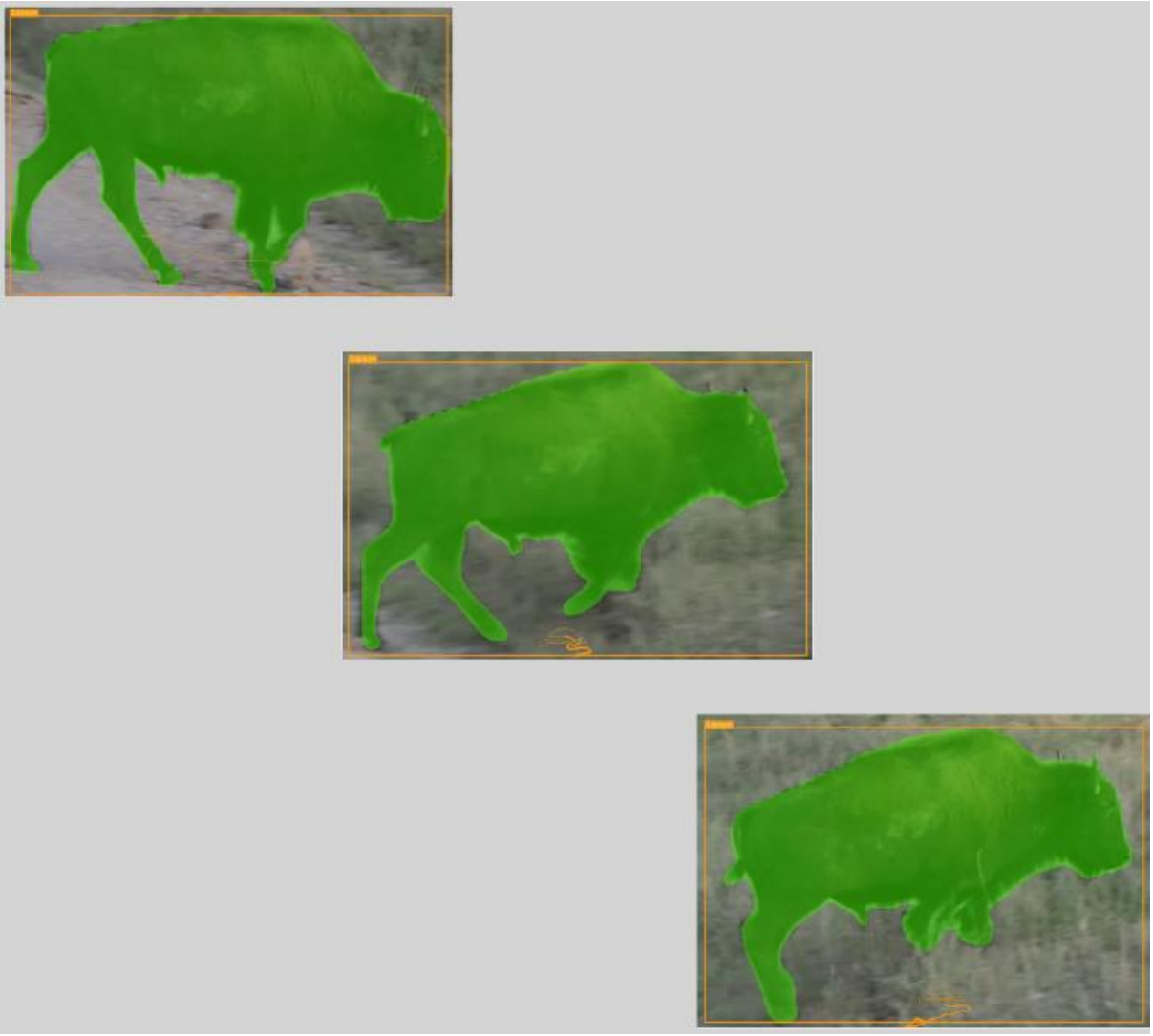

Figure 29 Samples results of the proposed animal tracking system, showing the tracking of a bison.

4.3 Experimental Results for the Proposed Speed Estimation Technique

The estimated speed for several pedestrians is shown in Table 3. These results are in line with the average walking speed of a human, as the slower pedestrian was an elderly woman, and the two faster pedestrian were adult men, with one walking at a slightly faster pace than the other.

Table 7 Results of the proposed speed estimation technique.

Pedestrian	Elapsed Time (s)	m/s	Distance (m)	Mph	Kmph
1	4	2.875	3.5	1.957	3.15
2	3	3.833	3.5	2.6	4.2
3	2	6.3	3.5	3.914	6.3

4.4 Experimental Results for the Proposed Trajectory Prediction Technique

Similarly to the comparison between members of the YOLO family of OD algorithms, another set of experiments was conducted to determine the most suitable ML regression model to perform trajectory prediction. The following regression models were compared: Partial Least Squares (PLS), AdaBoost, Bagging, ExtraTrees, Gradient Boosting, Random Forest, Stacking, Voting, Histogram-Based Gradient Boosting, Isotonic, Ridge, Stochastic Gradient Descent (SGD), Linear, ElasticNet, Least Angle Regression (LARS), LassoLARS, ARD, Bayesian Ridge, k-NN, Radius Neighbors, Nu Support Vector Regression (NuSVR), SVR, and Decision Tree. The results of these experiments are shown in Table 6.

It was determined that the k-NN regression model was the most accurate for trajectory prediction. This approach has an MSE of 0.019195935 and an R^2 score of 0.980804065. The accuracy of the k-NN regressor in accurately predicting the trajectory of a pedestrian is shown in Figure 30.

Table 6 Comparison of ML regressors for trajectory prediction.

Regressor	**MSE**	**R^2**
SGD	0.456198266	0.543801734
ElasticNet	0.452185514	0.547814486
LassoLARS	0.43861302	0.56138698

Linear	0.438202932	0.561797068
LARS	0.438202932	0.561797068
Ridge	0.438139723	0.561860277
ARD	0.437786109	0.562213891
PLS	0.430287031	0.569712969
BayesianRidge	0.430045613	0.569954387
HistGradientBoosting	0.167656004	0.832343996
Radius Neighbor	0.074559773	0.925440227
NuSVR	0.070928902	0.929071098
SVR	0.059189278	0.940810722
Bagging	0.057164319	0.942835681
Voting	0.056655007	0.943344993
AdaBoost	0.042503271	0.957496729
DT	0.038852635	0.961147365
Stacked	0.036594574	0.963405426
Gradient Boosting	0.035567388	0.964432612
Random Forest	0.035467849	0.964532151
ExtraTrees	0.023344438	0.976655562
Isotonic	0.023119719	0.976880281
k-NN	0.019195935	0.980804065

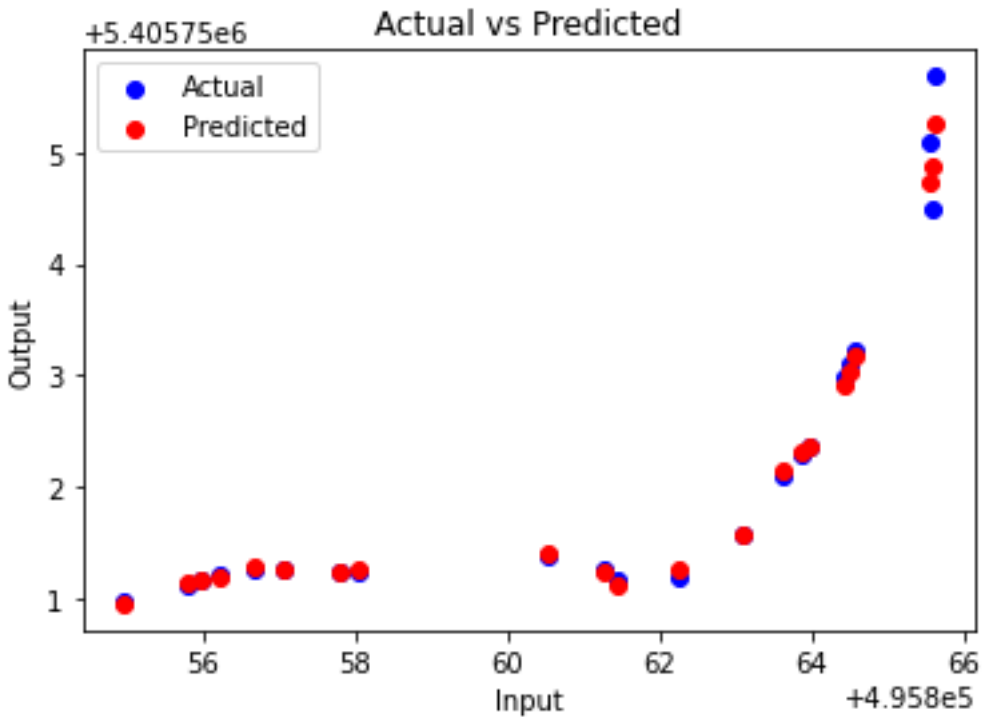

Figure 30 The actual position of a pedestrian compared to their k-NN regressor predicted position.

CHAPTER V: CONCLUSION AND FUTURE RESULTS

5.1 Conclusion

This thesis proposed techniques for a pedestrian detection and tracking system for AVs that differentiated between adult and child pedestrians. This system was then expanded to also detect and track 51 classes of commonly encountered animals around the world. In addition to the detection and tracking of these objects, a novel speed estimation technique was developed to estimate the speed of these objects, and trajectory prediction algorithms were implemented to make the system even more robust.

For the problem of differentiating between adults and children, a custom dataset was developed that was meant to address the lack of adequate data for this purpose in existing AV datasets. This custom dataset underwent preprocessing where the images were auto oriented, and was also augmented using mosaic augmentation, to make the model less susceptible to missed detections due to object occlusion. Once the custom dataset was developed, a custom implementation of the YOLOv8 algorithm with instance segmentation added was trained, and the subsequent weights file that was generated was used to train the DeepSORT OD model.

For the animal detection and tracking system, another custom dataset was developed, aiming to be the most robust dataset for animal detection on the roads available. This dataset consisted of samples from 51 types of mammals, birds, amphibians, and reptiles. This dataset was then used in a training process identical to the one used for pedestrian detection and tracking.

A novel speed estimation mathematical model based on the elapsed time of an object entering and exiting an area was devised. The novelty of this speed estimation technique comes down to the fact that it aims to provide a more accurate measure of the estimated speed compared to existing methods, as it aims to address issues related to the processing speed of the computer in determining accurate speed of the tracked object. In addition to the

speed estimation technique, the k-NN ML regressor was used to predict the trajectory of the objects.

5.2 Future Work

For future work on this proposed OD and OT system for AVs, further testing needs to be done to numerically evaluate the DeepSORT tracking model. Further consideration could also be paid to the speed estimation technique, to ensure it is even more accurate in estimating the speed of detected objects. The k-NN regression model used for trajectory prediction needs to be analyzed further, to see how it performs with different shapes of position data. Research could also be done to see how the k-NN model could be combined with other regressors or techniques for better performance.

Future work could also include implementing an approach for tasks such as lane detection, road sign detection, and collision prediction and prevention. Another interesting avenue that could be explored is methods to teach an AV to try and predict future events based on certain scene characteristics, for example, if there is a sports ball rolling across the road, it could be likely that a child is close behind, and the AV needs to be more aware. Another consideration could be analyzing detection and tracking performance based on the gender of the detected pedestrian.

REFERENCES

Andrie Asmara, R., Ridwan, M., & Budiprasetyo, G. (2021). Haar cascade and convolutional neural network face detection in client-side for cloud computing face recognition. *2021 International Conference on Electrical and Information Technology (IEIT)*.

Autonomous Vehicles Market Size & Share Report, 2030. (n.d.). Retrieved March 9, 2023, from https://www.grandviewresearch.com/industry-analysis/autonomous-vehicles-market

Bagloee, S. A., Tavana, M., Asadi, M., & Oliver, T. (2016). Autonomous vehicles: Challenges, opportunities, and future implications for transportation policies. *Journal of Modern Transportation*, *24*(4), 284–303.

Balbuzanov, T. G., & Evstatiev, B. I. (2019). Pedestrian presence detection system based on image processing. *2019 IEEE 25th International Symposium for Design and Technology in Electronic Packaging (SIITME)*.

Balntas, V., Johns, E., Tang, L., & Mikolajczyk, K. (2016). PN-Net: Conjoined triple deep network for learning local image descriptors. [Preprint]. *arXiv*.

Barnes, D., Gadd, M., Murcutt, P., Newman, P., & Posner, I. (2020). The oxford radar robotcar dataset: A radar extension to the Oxford RobotCar Dataset. *2020 IEEE International Conference on Robotics and Automation (ICRA)*.

Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. (2016). Simple online and realtime tracking. *2016 IEEE International Conference on Image Processing (ICIP)*.

Bochkovskiy, A., Wang, C. Y., & Mark Liao, H. Y. (2020). YOLOv4: Optimal speed and accuracy of object detection. [Preprint]. *arXiv*.

Brandão, M. (2019). Age and gender bias in pedestrian detection algorithms [Preprint]. *arXiv*.

Clay, D. (1995). Driver attitude and attribution: Implications for accident prevention (Doctoral dissertation). Cranfield University, School of Engineering, Cranfield, UK.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*.

Davis, J. W. (2001). Visual categorization of children and adult walking styles. *Lecture Notes in Computer Science*, 295–300.

Deng, J., Dong, W., Socher, R., & Li, L.-J. (2009). ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/cvpr.2009.5206848

Dollar, P., Wojek, C., Schiele, B., & Perona, P. (2009). Pedestrian detection: A benchmark. *2009 IEEE Conference on Computer Vision and Pattern Recognition*.

Dwyer, B., Nelson, J. (2022), Solawetz, J., et. al. Roboflow (Version 1.0) [Software]. Available from https://roboflow.com. computer vision.

Fagnant, D. J., & Kockelman, K. (2015). Preparing a nation for autonomous vehicles: Opportunities, barriers and policy recommendations. *Transportation Research Part A: Policy and Practice*, *77*, 167–181.

Famili, A., Shen, W.-M., Weber, R., & Simoudis, E. (1997). Data preprocessing and Intelligent Data Analysis. *Intelligent Data Analysis*, *1*(1), 3–23.

Favarò, F. M., Nader, N., Eurich, S. O., Tripp, M., & Varadaraju, N. (2017). Examining accident reports involving autonomous vehicles in California. *PLOS ONE*, *12*(9).

Felzenszwalb, P., McAllester, D., & Ramanan, D. (2008). A discriminatively trained, multiscale, deformable part model. *2008 IEEE Conference on Computer Vision and Pattern Recognition*.

Girshick, R. (2015). Fast R-CNN. *2015 IEEE International Conference on Computer Vision (ICCV)*.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*.

Goel, D., & Chen, T. (2007). Real-time pedestrian detection using eigenflow. *2007 IEEE International Conference on Image Processing*.

Ha, T., Kim, S., Seo, D., & Lee, S. (2020). Effects of explanation types and perceived risk on trust in Autonomous Vehicles. *Transportation Research Part F: Traffic Psychology and Behaviour*, *73*, 271–280.

Hafiz, A. M., & Bhat, G. M. (2020). A survey on instance segmentation: State of the art. *International Journal of Multimedia Information Retrieval*, *9*(3), 171–189.

Han, X., Leung, T., Jia, Y., Sukthankar, R., & Berg, A. C. (2015). MatchNet: Unifying feature and metric learning for patch-based matching. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Hans, W. J., Sherlin, V., & Venkateswaran, N. (2020). On-road deer detection for advanced driver assistance using Convolutional Neural Network. *International Journal of Advanced Computer Science and Applications*, *11*(4).

Hao, W., & Zhili, S. (2020). Improved mosaic: Algorithms for more complex images. *Journal of Physics: Conference Series*, *1684*(1), 012094.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(9), 1904–1916.

He, M., Luo, H., Chang, Z., & Hui, B. (2017). Pedestrian detection with semantic regions of interest. *Sensors*, *17*(11), 2699.

Heo, M.-H., Lee, J., Shin, D.-J., Lee, Y.-S., & Kim, J.-J. (2021). Pedestrian recognition technology using YOLO. Turkish Online Journal of Qualitative Inquiry, 12(6), 1662-1666.

Holland, C., & Hill, R. (2007). The effect of age, gender and driver status on pedestrians' intentions to cross the road in risky situations. *Accident Analysis & Prevention*, *39*(2), 224–237.

Huang, X., Cheng, X., Geng, Q., Cao, B., Zhou, D., Wang, P., Lin, Y., & Yang, R. (2018). The Apolloscape dataset for autonomous driving. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*.

Ilas, C. (2013). Electronic Sensing Technologies for autonomous ground vehicles: A Review. *2013 8th International Symposium on Advanced Topics in Electrical Engineering (ATEE)*.

Ince, O. F., Ince, I. F., Park, J. S., Song, J., & Yoon, B. W. (2017). Child and adult classification using biometric features based on video analytics. *ICIC Express Letters Part B: Applications*, 8(5), 819-825.

Ince, O. F., Park, J. S., Song, J., & Yoon, B. W. (2014). Child and adult classification using ratio of head and body heights in images. *International Journal of Computer and Communication Engineering*, 3(2), 120-122.

Ince, O. F., Yildirim, M. E., Park, J. S., & Yoon, B. W. (2015). Video based adult and child classification by using body proportion. *2015 The 5th International Workshop on Computer Science and Engineering – Information Processing and Control Engineering*.

Jaarsma, C. F., van Langevelde, F., & Botma, H. (2006). Flattened fauna and mitigation: Traffic victims related to road, traffic, vehicle, and species characteristics. *Transportation Research Part D: Transport and Environment*, *11*(4), 264–276.

Janai, J., Güney, F., Behl, A., & Geiger, A. (2020). Computer Vision for autonomous vehicles: Problems, datasets and state of the art. *Foundations and Trends® in Computer Graphics and Vision*, *12*(1–3), 1–308.

Jocher, G. (2020). YOLOv5 by Ultralytics (Version 7.0) [Computer software]. https://doi.org/10.5281/zenodo.3908559

Jocher, G., Chaurasia, A., & Qiu, J. (2023). YOLO by Ultralytics (Version 8.0.0) [Computer software]. https://github.com/ultralytics/ultralytics

Kahlon, G. S., Singh, H., Saini, M., & Kaur, S. (2023). An intelligent framework to detect and generate alert while cattle lying on road in dangerous states using surveillance videos. *Multimedia Tools and Applications*.

Knutson, R. M. (2006). Flattened fauna, revised: A field guide to common animals of roads, streets, and highways. *Ten Speed Press*.

Kogure, S., Watabe, K., Yamada, R., Aoki, Y., Nakamura, A., & Kataoka, H. (2022). Age should not matter: Towards more accurate pedestrian detection via self-training. *AAAI Workshop on Artificial Intelligence with Biased or Scarce Data (AIBSD)*.

Lee, S., Kim, Y., Kahng, H., Lee, S.-K., Chung, S., Cheong, T., Shin, K., Park, J., & Kim, S. B. (2020). Intelligent Traffic Control for Autonomous Vehicle Systems based on machine learning. *Expert Systems with Applications, 144*, 113074.

Levering, A., Tomko, M., Tuia, D., & Khoshelham, K. (2021). Detecting unsigned physical road incidents from driver-view images. *IEEE Transactions on Intelligent Vehicles, 6*(1), 24–33.

Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., Ke, Z., Li, Q., Cheng, M., Nie, W., Li, Y., Zhang, B., Liang, Y., Zhou, L., Xu, X., Chu, X., Wei, X., Wei, X. (2022). YOLOv6: A single stage object detection framework for industrial applications. [Preprint]. *arXiv*.

Lin, J. M., Lin, W. L., & Fan, C. P. (2022). Age group classifier of adults and children with YOLO-based deep learning pre-processing scheme for embedded platforms. *2022 IEEE 12th International Conference on Consumer Electronics (ICCE-Berlin).*

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft Coco: Common Objects in Context. *Computer Vision – ECCV 2014*, 740–755.

Luo, W., Xing, J., Milan, A., Zhang, X., Liu, W., & Kim, T.-K. (2021). Multiple object tracking: A literature review. *Artificial Intelligence*, *293*, 103448.

Maddern, W., Pascoe, G., Linegar, C., & Newman, P. (2016). 1 year, 1000 km: The Oxford Robotcar Dataset. *The International Journal of Robotics Research*, *36*(1), 3–15.

Manikandan, N. S. & Ganesan, K. (2019). Deep learning based automatic video annotation tool for self-driving car. [Preprint]. a*rXiv*.

Matuska, S., Hudec, D., Kamencay, P., & Trnovsky, T. (2016). A video camera road sign system of the early warning from collision with the wild animals. *Civil and Environmental Engineering*, 12(1), 42-46.

Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N., & Terzopoulos, D. (2021). Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7), 3523-3542.

Mohideen Meera Sha, U. K. (2021). Pedestrian multiple object tracking using deep learning. (Master's thesis). Uppsala University, Department of Information Technology, Uppsala, Sweden.

Mowen, D., Munian, Y., & Alamaniotis, M. (2022). Improving road safety during nocturnal hours by characterizing animal poses utilizing CNN-based analysis of thermal images. *Sustainability*, *14*(19), 12133.

O'Shea, K. & Nash, R. (2015). An introduction to convolutional neural networks. [Preprint]. *arXiv*.

Prabhakar, G., Kailath, B., Natarajan, S., & Kumar, R. (2017). Obstacle detection and classification using Deep Learning for tracking in high-speed autonomous driving. *2017 IEEE Region 10 Symposium (TENSYMP)*.

Protopapadakis, E., Katsamenis, I., & Doulamis, A. (2020). Multi-label deep learning models for continuous monitoring of road infrastructures. *Proceedings of the 13th ACM International Conference on Pervasive Technologies Related to Assistive Environments*.

Rasouli, A., Kotseruba, I., & Tsotsos, J. K. (2017). Agreeing to cross: How drivers and pedestrians communicate. *2017 IEEE Intelligent Vehicles Symposium (IV)*.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Redmon, J., & Farhadi, A. (2017). Yolo9000: Better, faster, stronger. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr.2017.690

Redmon, J., & Farhadi, A. (2018). YOLOv3: An incremental improvement. [Preprint]. *arXiv*.

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6), 1137–1149.

Reyes-García, C. A., Morales-Vargas, E., Peregrina-Barreto, H., & Manfredi, C. (2019). Discrimination between children and adult faces using body and head ratio and geometric features. *Models and Analysis of Vocal Emissions for Biomedical Applications: 11th International Workshop*.

Rosenfeld, A. (1988). Computer vision: Basic principles. *Proceedings of the IEEE*, *76*(8), 863–868.

Santhanam, S., B, S. S., Panigrahi, S. S., Kashyap, S. K., & Duriseti, B. K. (2021). Animal detection for road safety using Deep Learning. *2021 International Conference on Computational Intelligence and Computing Applications (ICCICA)*.

Sarmento, A., Garcia, B., Coriteac, L., & Navarenho, L. (2017). The autonomous vehicle challenges for emergent market. *SAE Technical Paper Series*.

Sedgh Gooya, E., Aram, F., Kaddah, W., Elbouz, M., & Alfalou, A. (2021). A human body morphology detector: A distinctive filter to differentiate the image-based process depending on whether the person is a child or an adult. *Pattern Recognition and Tracking XXXII*.

Sharma, D., Hade, T., & Tian, Q. (2022). Comparison Of Deep Object Detectors On A New Vulnerable Pedestrian Dataset [Preprint]. *arXiv*.

Sharma, S. U., & Shah, D. J. (2017). A practical animal detection and collision avoidance system using computer vision technique. *IEEE Access*, *5*, 347–358.

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for Deep Learning. *Journal of Big Data*, *6*(1).

Simo-Serra, E., Trulls, E., Ferraz, L., Kokkinos, I., Fua, P., & Moreno-Noguer, F. (2015). Discriminative learning of deep convolutional feature point descriptors. *2015 IEEE International Conference on Computer Vision (ICCV)*.

Smith, A. (2019). *3. Americans' attitudes toward driverless vehicles*. Pew Research Center: Internet, Science & Tech. Retrieved March 9, 2023, from https://www.pewresearch.org/internet/2017/10/04/americans-attitudes-toward-driverless-vehicles/.

Soleimanitaleb, Z. & Keyvanrad, M. A. (2022). Single object tracking: A survey of methods, datasets, and evaluation metrics. [Preprint]. *arXiv*.

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of Simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*.

Wang, C. Y., Bochkovskiy, A., Mark Liao, H. Y. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. [Preprint]. *arXiv*.

Wang, J., Zhang, L., Huang, Y., Zhao, J., & Bella, F. (2020). Safety of Autonomous vehicles. *Journal of Advanced Transportation, 2020,* 1–13.

Wojke, N., Bewley, A., & Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. *2017 IEEE International Conference on Image Processing (ICIP)*.

Yu, F., Chen, H., Wang, X., Xian, W., Chen, Y., Liu, F., Madhavan, V., & Darrell, T. (2020). BDD100K: A diverse driving dataset for heterogeneous multitask learning. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yudin, D., Sotnikov, A., & Krishtopik, A. (2019). Detection of big animals on images with road scenes using Deep Learning. *2019 International Conference on Artificial Intelligence: Applications and Innovations (IC-AIAI)*.

Zagoruyko, S. & Komodakis, N. (2015). Learning to compare image patches via convolutional neural networks. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zou, Z., Chen, K., Shi, Z., Guo, Y., & Ye, J. (2023). Object detection in 20 years: A survey. *Proceedings of the IEEE, 111*(3), 257–276.