Detekcija orijentiranih objekata u termovizijskim zračnim slikama s dronova temeljena na sintetičkim podacima i neuronskim mrežama

Jurasović, Andreja

Master's thesis / Diplomski rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Electrical Engineering and Computing / Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:168:697473

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

MASTER THESIS No. 79

ORIENTED OBJECT DETECTION IN THERMAL AERIAL IMAGES FROM DRONES BASED ON SYNTHETIC DATA AND NEURAL NETWORKS

Andreja Jurasović

Zagreb, June 2024

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

MASTER THESIS No. 79

ORIENTED OBJECT DETECTION IN THERMAL AERIAL IMAGES FROM DRONES BASED ON SYNTHETIC DATA AND NEURAL NETWORKS

Andreja Jurasović

Zagreb, June 2024

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Zagreb, 04 March 2024

MASTER THESIS ASSIGNMENT No. 79

Student:	**Andreja Jurasović (0036525063)**
Study:	Information and Communication Technology
Profile:	Control Systems and Robotics
Mentor:	prof. Stjepan Bogdan

Title: **Oriented Object Detection in Thermal Aerial Images from Drones Based On Synthetic Data and Neural Networks**

Description:

The first phase of work on the thesis is dedicated to review of the literature on thermal imaging data sets from an aerial perspective and, the literature on neural networks for oriented objects. After that, the 3D thermal model of the multirotor unmanned aerial vehicle should be devised. This model should be used for generation of a diverse set of data using existing automated procedures, which should be combined with the thermal backgrounds recorded from the aerial vehicle using a thermal camera. In the next step, it is necessary to generate annotations in the form of an oriented bounding rectangle and to combine synthetic annotations with background image annotations. The newly created data set should be used for training of neural networks for the detection of oriented objects, followed by an investigation on the ratio of accuracy and performance speed of neural networks. In the last phase of work on the thesis, it is necessary to collect a smaller set of real data, which contains all the classes included in this work, and to validate neural networks on that set.

Submission date: 28 June 2024

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 79

Pristupnica:	**Andreja Jurasović (0036525063)**
Studij:	Informacijska i komunikacijska tehnologija
Profil:	Automatika i robotika
Mentor:	prof. dr. sc. Stjepan Bogdan

Zadatak: **Detekcija orijentiranih objekata u termovizijskim zračnim slikama s dronova temeljena na sintetičkim podacima i neuronskim mrežama**

Opis zadatka:

U prvoj fazi rada na zadatku potrebno je napraviti pregled literature o termovizijskim skupovima podatka iz zračne perspektive i neuronskim mrežama za orijentirane objekte. Nakon toga je potrebno u 3D alatu za računalnu grafiku modelirati termalni model multirotorske bespilotne letjelice te koristeći postojeće automatizirane postupke, generirati raznovrsan skup podataka, koji se treba spojiti s termalnim pozadinama snimljenim s letjelice pomoću termalne kamere. U sljedećem koraku potrebno je generirati anotacije u obliku orijentiranog omeđujućeg pravokutnika te spojiti sintetičke anotacije s anotacijama pozadinskih slika. Na novonastalom skupu podataka trenirati neuronske mreže za detekciju orijentiranih objekata te istražiti omjer točnosti i brzine izvođenja neuronskih mreža. U zadnjoj fazi rada na zadatku potrebno je prikupiti manji skup realnih podataka, koji sadrži sve klase obuhvaćene ovim radom te validirati neuronske mreže na tom skupu.

Rok za predaju rada: 28. lipnja 2024.

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

MASTER THESIS No. 79

ORIENTED OBJECT DETECTION IN THERMAL AERIAL IMAGES FROM DRONES BASED ON SYNTHETIC DATA AND NEURAL NETWORKS

Andreja Jurasović

Zagreb, July, 2024

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 79

ORIJENTIRANA DETEKCIJA OBJEKATA NA TERMALNIM SLIKAMA IZ BESPILOTNIH LETJELICA NA TEMELJU SINTETIČKIH PODATAKA I NEURONSKIH MREŽA

Andreja Jurasović

Zagreb, srpanj, 2024.

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Zagreb, 04 March 2024

MASTER THESIS ASSIGNMENT No. 79

Student:	**Andreja Jurasović (0036525063)**
Study:	Information and Communication Technology
Profile:	Control Systems and Robotics
Mentor:	prof. Stjepan Bogdan

Title: **Oriented Object Detection in Thermal Aerial Images from Drones Based On Synthetic Data and Neural Networks**

Description:

The first phase of work on the thesis is dedicated to review of the literature on thermal imaging data sets from an aerial perspective and, the literature on neural networks for oriented objects. After that, the 3D thermal model of the multirotor unmanned aerial vehicle should be devised. This model should be used for generation of a diverse set of data using existing automated procedures, which should be combined with the thermal backgrounds recorded from the aerial vehicle using a thermal camera. In the next step, it is necessary to generate annotations in the form of an oriented bounding rectangle and to combine synthetic annotations with background image annotations. The newly created data set should be used for training of neural networks for the detection of oriented objects, followed by an investigation on the ratio of accuracy and performance speed of neural networks. In the last phase of work on the thesis, it is necessary to collect a smaller set of real data, which contains all the classes included in this work, and to validate neural networks on that set.

Submission date: 28 June 2024

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 79

Pristupnica:	**Andreja Jurasović (0036525063)**
Studij:	Informacijska i komunikacijska tehnologija
Profil:	Automatika i robotika
Mentor:	prof. dr. sc. Stjepan Bogdan

Zadatak: **Detekcija orijentiranih objekata u termovizijskim zračnim slikama s dronova temeljena na sintetičkim podacima i neuronskim mrežama**

Opis zadatka:

U prvoj fazi rada na zadatku potrebno je napraviti pregled literature o termovizijskim skupovima podatka iz zračne perspektive i neuronskim mrežama za orijentirane objekte. Nakon toga je potrebno u 3D alatu za računalnu grafiku modelirati termalni model multirotorske bespilotne letjelice te koristeći postojeće automatizirane postupke, generirati raznovrsan skup podataka, koji se treba spojiti s termalnim pozadinama snimljenim s letjelice pomoću termalne kamere. U sljedećem koraku potrebno je generirati anotacije u obliku orijentiranog omeđujućeg pravokutnika te spojiti sintetičke anotacije s anotacijama pozadinskih slika. Na novonastalom skupu podataka trenirati neuronske mreže za detekciju orijentiranih objekata te istražiti omjer točnosti i brzine izvođenja neuronskih mreža. U zadnjoj fazi rada na zadatku potrebno je prikupiti manji skup realnih podataka, koji sadrži sve klase obuhvaćene ovim radom te validirati neuronske mreže na tom skupu.

Rok za predaju rada: 28. lipnja 2024.

I would like to thank my family and friends who supported me during my master studies, especially Josip and Nika for their greatest support. Thanks to the mentor, prof. Ph.D. Stjepan Bogdan and assistant Antonella Barišić Kulaš, without whom this work would not have been possible.

Contents

1 Introduction

1.1 Thermal imaging datasets from an aerial perspective

Thermal images, also known as thermograms, are visual displays of the amount of infrared energy emitted, transmitted, and reflected by objects. The human eye and conventional cameras capture only visible light, while infrared energy is part of the electromagnetic spectrum. It is characterized by wavelengths longer than visible light but shorter than microwave radiation. Thermal cameras detect this energy and convert it to an image that displays the temperature distribution of a scene, assigning different temperatures to various color codes. These are typically seen as grayscale or color maps. Thermal images are visualization of scenes invisible to the naked eye, which is particularly useful in low-light or obscured conditions. In scenarios where visibility is diminished, such as during nighttime or through smoke and fog, capturing images using only visible light is problematic. This is exactly where thermal imaging gives remarkable results. Thermal imaging is based on the principle that all objects above absolute zero emit thermal infrared radiation as a function of their temperature. This principle is based on Planck's law of blackbody radiation, which describes the spectral density of electromagnetic radiation emitted by a black body in thermal equilibrium at a given temperature. Thermal cameras are equipped with sensors that detect this radiation and translate it into an electrical signal, which is then processed to form a digital image. Each pixel of a thermal image corresponds to the infrared radiation from a specific part of the imaged area which translates into a temperature value. This makes thermal images particularly useful for detecting variations in temperature across a scene. Aerial surveillance technologies specifically may give vital information frequently unavailable through traditional means and so they have become essential to modern security, search and rescue, environmental

monitoring and military activities. Unmanned aerial vehicles (UAVs) offer a versatile and dynamic approach for capturing data — they excel in covering large areas rapidly, accessing hard-to-reach locations, reducing human risk in dangerous environments and delivering real-time data for quick decision-making. Integration of thermal imaging into drones amplifies their effectiveness, enabling the detection of heat signatures from above with precision and reliability. By giving law enforcement and emergency response teams a tool that can function around-the-clock in a variety of environments, this combination enhances situational awareness and also broadens their operational capabilities as well.

Figure 1.1: Example of thermal image

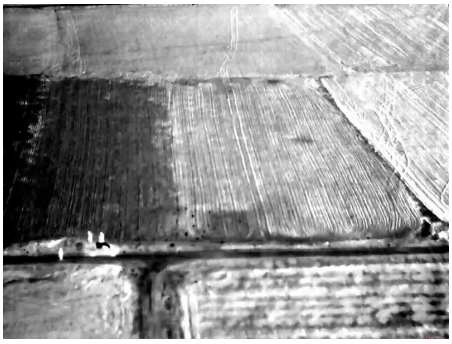

Figure 1.2: Example of thermal image

1.2 Neural networks for object detection

Neural networks atre a subset of machine learning models. They have already shown an outstanding performance on image classification tasks so it was logical to take further step and use them for object detection. This extends not only to classifying, but also localizing different class objects.

In context of object detection, neural networks can be categorized into two main types: one-stage detectors and two-stage detectors. One-stage detectors simplify the pro-

cess of object detection by directly predicting the bounding boxes and class probabilities in one evaluation from full images. This is how they optimize the speed and efficiency of the detection. SSD (Single Shot Multibox Detector) is a one-staged detector that achieves a good balance between speed and accuracy. To predict the object at multiple scales and aspect ratios, it uses a series of convolutional layers at different scales. DETR (Detection Transformer) treats object detection as a direct set prediction problem. The transformer that model uses is for doing global reasoning across the entire image and outputs a fixed number of predictions, where each prediction directly corresponds to a final detected object. DETR eliminates the need for many hand-designed components, which are common in other detectors. It is used for its simplicity and effectiveness, especially in handling complex scenes with a clear global view. On the other side, two-stage detectors process the detection in two phases. The first one is region proposal, where a set of candidates for object's bounding boxes is generated. Those candidates are called 'region proposals' because they indicate areas of the image where there is a high likelihood of containing an object. The second phase is classification and bounding box regression, where each proposed region is analyzed to find out the class of the object contained within it and to refine the bounding box coordinates for a precise fit. The example of a two-stage detector is a faster R-CNN with its high accuracy. It builds on earlier models like R-CNN and Fast R-CNN with the RPN (Region Proposal Network) which is a fully convolutional network that predicts object bounds and object's scores at each position. It is trained end-to-end to generate high-quality region proposals, which are then used by a Fast R-CNN detector for the final classification and bounding box regression. This model can be slower than one-stage detectors because of the two-steps processing. However, it is highly accurate and robust, especially in dealing with complex images where precise detection is crucial, for example in inspection systems or medical imaging. On the other hand, one-stage detectors are faster, which makes suitable them for applications requiring real-time performance like autonomous driving or video surveillance. DETR offers a more unified approach with potentially lower engineering complexity and better scalability for different tasks.

Over the years, this area has seen significant advancements, driven by improvements in neural network architectures, training methodologies, and the availability of large-scale annotated datasets. YOLO (You Only Look Once) framework further revolution-

ized object detection by framing it as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. Its iterations, especially YOLOv3 and YOLOv4, have focused on improving accuracy, speed, and the ability to detect small objects. Great strides in this field were achieved through architectural tweaks and advanced training techniques. With the challenge of obtaining large, annotated datasets, several studies have explored the use of synthetic data generation and data augmentation techniques to expand training datasets artificially. This enhances the diversity of training samples and, even more vital, helps in training more robust models capable of generalizing well across different real-world scenarios.

2 The Datasets

2.1 Background images

2.1.1 Synthetic Dataset of Urban Environment

The HIT-UAV10 dataset represents a development in the field of unmanned aerial vehicle (UAV)-based object detection, focusing on high-altitude infrared thermal imaging. This dataset comprises 2,898 carefully collected infrared thermal images, which have been extracted from a larger pool of 43,470 frames spanning hundreds of videos. These images have been captured across a wide variety of urban and semi-urban environments: schools, parking lots, roads and playgrounds, ensuring a broad representation of real-world scenarios. The HIT-UAV10 dataset is designed to enhance research in UAV-based object detection and so it records crucial flight information for each captured image. This information includes the UAV's flight altitude, camera perspective, the intensity of daylight at the time of capture, and the exact date of the image shoot. Such comprehensive metadata is important for exploring the impact of varying flight conditions on the accuracy of object detection algorithms. That is why this dataset covers flight altitudes ranging from 60 to 130 meters and camera angles from 30 to 90 degrees, allowing researchers to identify the influence of these factors on the detection efficacy. Oriented bounding boxes more tightly encloses object so that less backgrund is contained in the bounding box, therefore location is more precise and gives another information about the object orientation in the two dimensions. Oriented bounding boxes are particularly useful for precise object localization in complex aerial views. In these cases, objects may not only overlap but also appear at various angles, making standard bounding boxes less effective. The objects annotated in this dataset include high-frequency urban elements such as persons, cars, bicycles and other vehicles. These are chosen for their relevance to typical UAV operational tasks such as traffic monitoring, urban planning and emer-

gency response. Annotations have been applied to ensure high reliability in object identification, crucial for training robust neural networks. Moreover, the data was captured across various times of day and under varying light conditions, providing the dataset a robust foundation for developing object detection systems effective under a wide range of operational scenarios and diverse conditions. This provides a particular advantage to applications that require nighttime operation, where infrared thermal imaging is indispensable. The HIT-UAV10 dataset enables the development and refinement of advanced object detection algorithms by providing a rich source of high-quality, annotated thermal images.

Figure 2.1: Background image at 30 degrees

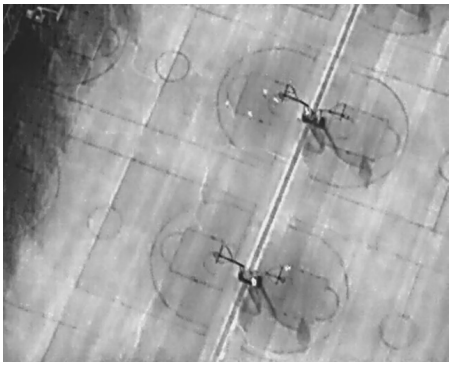

Figure 2.2: Background image at 70 degrees

2.1.2 Synthetic Dataset of Rural Environment

The MONET dataset is a multimodal drone thermal dataset designed for studying object localization and behavior in rural environments. This dataset consists of approximately 53,000 frames enriched with 162,000 manually annotated bounding boxes, highlighting human and vehicle activities captured through a thermal camera mounted on a drone. The recordings encapsulate two distinct rural settings, each with unique scene structures

and cluttered backgrounds providing a challenging diversity for object detection models. Two distinct rural settings are two groups of images. One group contains images taken in environment close to the rural road and other group contains images taken in environment close to the runway. One idea of the thesis is to easily compare two generated datasets, so not all thermal images from the MONET dataset were used. Thermal images that were used were the ones taken under the camera angles from 30 to 90 degrees, matching the camera angles from the HIT-UAV10 dataset. This makes a total of 2716 images that were used as backgrounds for rendered images. Each image in the MONET dataset is timestamp-aligned with extensive drone metadata. This includes attitudes, speed, altitude and GPS coordinates, crucial for detailed environmental and analytical studies. MONET's detailed annotations encompass three primary categories: persons, vehicles and a region marked as 'ignore' to avoid algorithmic bias towards non-target areas. The main advantage of MONET dataset is its focus on providing a richly annotated dataset that supports object detection, as well as nuanced understanding of object behaviors from different and moving viewpoints. This makes MONET highly suitable for advancing research in drone-based surveillance, agricultural monitoring, and rural security applications, where understanding and reacting to dynamic object interactions is crucial.

Figure 2.3: Background image at 50 degrees

2.2 Thermal Models

The creation of thermal models using Blender, a comprehensive open-source 3D modeling software, is the next logical step in context of advanced visualization and simulation for thermal imaging applications. This process involves the design and refinement of

Figure 2.4: Background image at 70 degrees

3D models that simulate the thermal properties and behaviors of real-world objects and environments under various conditions. Blender is a facility for real-world simulation of thermal characteristics by allowing for the integration of material properties that emulate thermal emissivity and reflectivity, both essential for creating realistic thermal imagery. As mentioned, this is particularly useful in scenarios where actual thermal data is difficult to obtain or when researchers wish to study the thermal behavior of objects in controlled, hypothetical or extreme conditions not easily replicable in the real world.

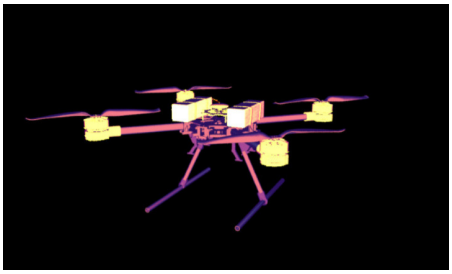

Figure 2.5: Thermo model of the UAV

Figure 2.6: Thermo model of a deer

2.2.1 Thermal Drone

UAV 3D thermal model was modified in the Shading part of the Blender to achieve its thermal image. Thermal coloring is managed with different nodes. First, there is a main group of nodes to get the infrared effect containing these nodes: RGB, Diffuse BSDF, Glossy BSDF, Mix Shader, subgroup Infreredeffct and final one, Material Output. This is how they work together:

1. **Diffuse BSDF Node**

   This node is essential to simulate the way surfaces scatter light. It provides a non-reflective, matte surface which reflects light uniformly in all directions. In this setup, the color input is driven by an RGB node, allowing for precise color specification, which affects how the material absorbs and scatters light.

2. **Glossy BSDF Node**

   In contrast to the diffuse shader, simulates the reflection of light in a more concentrated manner, like shiny or metallic surfaces do. This node's parameters roughness and the Beckmann distribution model control the sharpness and spread of the specular highlights on the surface.

3. **Mix Shader Node**

This node blends the outputs of the Diffuse and Glossy BSDF shaders. The factor controlling the mix is set to 0.500, indicating an equal contribution from both shaders. The result is a surface with both characteristics: matte from the diffuse component and shiny from the glossy component, which represent materials like semi-gloss plastics.

4. **Custom Node Group (*InfraredEffect*)**

   This node group represents a custom shader to modify the surface appearance and simulate an infrared view. It adjusts the material properties based on how they emit, absorb, or reflect infrared radiation.
   This is how the subgroup is structured:

   (a) **Texture Coordinate and Mapping Nodes**

   These nodes are used to control the placement and orientation of textures on the object. The Texture Coordinate node provides diverse options for how textures are mapped to the object, such as using object coordinates, UV maps, or generated coordinates. The Mapping node can adjust these textures by scaling, rotating, or translating them. The result is precise control over how the texture appears on the object.

   (b) **Image Texture Nodes**

   These nodes are linked to image textures that represent different material properties under thermal imaging. Each Image Texture node is connected to a different component, influencing various aspects of the material, such as color, emissivity or reflectivity in the thermal spectrum.

   (c) **Mix Shader Nodes**

   These nodes blend multiple shaders together based on a specified factor. They combine different effects from the image textures, allowing for complex material appearances.

   (d) **Diffuse BSDF and Emission Nodes**

The Diffuse BSDF node simulates a basic Lambertian surface, which is typically not very shiny and reflects light uniformly in all directions. The Emission node is crucial in a thermal material setup as it simulates the material emitting light (or in this case, heat), which is what thermal cameras capture. Different strengths of emission simulates different temperatures of the material.

(e) **Shader to RGB Node**

This node is particularly useful in non-photorealistic applications as it converts shader outputs into RGB colors. This can be helpful for creating stylized effects or for specific technical applications where shader data needs to be repurposed for diverse types of visual outputs.

(f) **ColorRamp Node**

This node remaps the colors of a shader's output to a specific gradient, typically used to visually denote different temperatures in thermal imaging. For example, cooler temperatures are mapped to darker colors and hotter areas to brighter, more intense colors.

(g) **Material Output Node**

This is the final node in the setup, like mentioned before.

5. **Material Output Node**

This node is the final stage where the combined shader effects are output to render the material on the 3D model. It ensures that all the shader effects are compiled and represented in the final visual output in the 3D viewport or render.

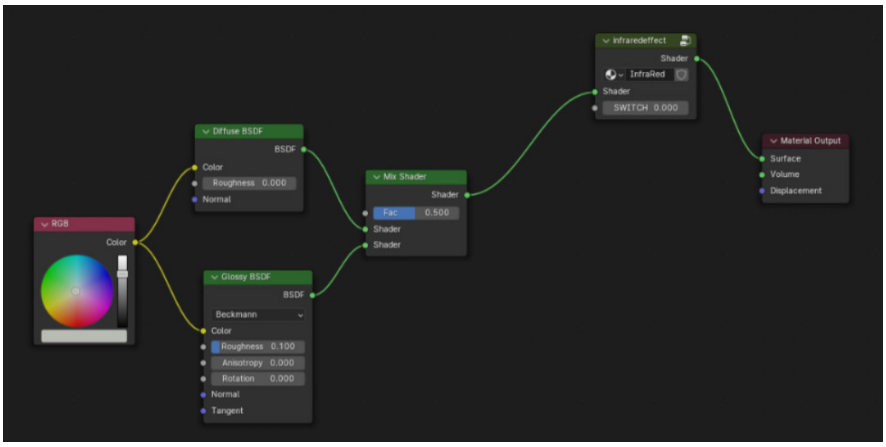

Figure 2.7: Blender nodes for thermal effect

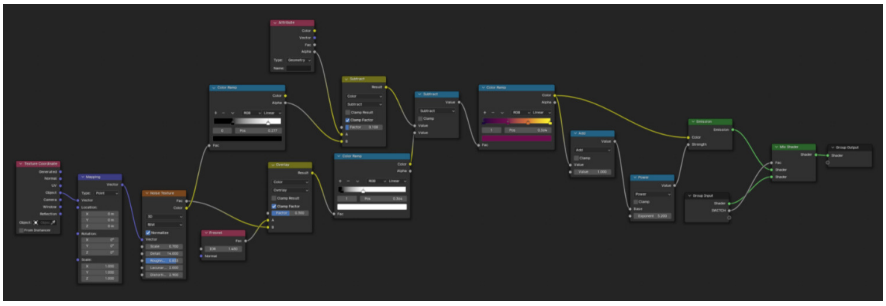

Figure 2.8: Subgroup of Blender nodes for thermal effect

2.2.2 Thermal Deer

Thermal shading of a deer is done based on the drone model with few adjustments because of the deer's materials, body hair, and antlers. The logic of the shading is the same as the one drone is using – a well-structured map of nodes creating the final thermal effect. This is how the main group is set:

1. **CarbonWave Node**

   This custom node is named for its texture or pattern output that simulates a carbon fiber weave. It is a base for the thermal texture. It outputs a pattern used to influence other properties in the material, such as color or emission strength, and is representative of varying material properties across the deer's body.

2. **Layer Weight Node**

   This node calculates the Fresnel effect, which describes how light reflects off surfaces at different angles. This is important in context of thermal imaging

because it is used to simulate how infrared radiation varies with angle. At the same time, it influences how various parts of the deer appear hotter or cooler based on their orientation relative to the camera.

3. **Mix Node (before Diffuse BSDF)**

   This node blends the CarbonWave texture with another unnamed input, using the Fresnel effect as the factor. This step adjusts the base color and emissive properties of the material based on the surface angle. It results in enhancing the realism of the thermal effect by varying the appearance according to the viewing perspective.

4. **Diffuse BSDF and Sheen BSDF Nodes**

   These shaders define the basic color and reflective properties of the surface under normal lighting conditions. The Diffuse BSDF provides a matte finish, while the Sheen BSDF adds a soft, velvet-like reflection, which is particularly useful for organic subjects like the deer itself to simulate the subtle glossiness found in natural fur.

5. **Mix Shader Nodes**

   These nodes combine the outputs of the previous shaders. The first Mix Shader blends the Diffuse and Sheen shaders linked to material properties or texture maps. The second Mix Shader then blends this result with the Glossy shader, controlled by a factor of 0.500, balancing between matte and glossy finishes to achieve a realistic surface under visible light.

6. ***InfraredEffect* Node**

   This custom node group is again crucial for simulating the thermal imaging effect. It adjusts the emissivity of the material, modifying how it emits infrared radiation.
   The *Infraredeffect* subgroup is the same as the drone's with just a few color adjustments to match the deer's temperature that differs from the drone's temperature.

7. **Final Mix Shader and Material Output**

   The last Mix Shader blends the visible light material properties with the *InfraredEffect* output, using another Fresnel factor (calculated inside the *InfraredEffect* node) to ensure that thermal properties are visible primarily from direct viewing angles. The Material Output node then finalizes the shader for rendering, displaying the combined effects of both visible and infrared properties on the model.

The deer's antlers shading is done the same as its body, with just color adjustment to match their temperature.

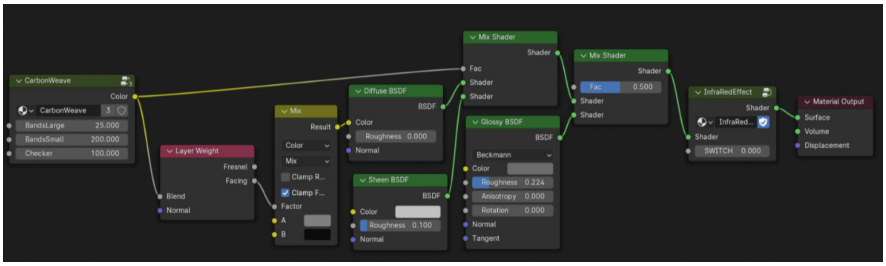

Figure 2.9: Blender nodes for thermal effect

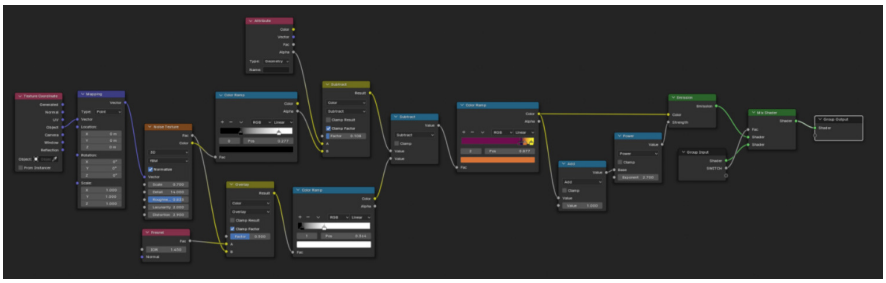

Figure 2.10: Subgroup of Blender nodes for thermal effect

2.3 Rendered Images

The final images are rendered in Blender. They are combinations of background images and thermal 3D Blender models. Two types of images are rendered – the first type is a combination of the HIT-UAV10 dataset and the 3D thermal model of the Eagle drone, and the second type is a combination of the MONET dataset and the 3D thermal model of the deer. For each background image, two final images are rendered to get the random position of the thermal object on the images and to randomize the final dataset.

These are the rendering settings used for both drone and deer models:

1. **Rendering Engine**

   The script specifies Blender's Cycles engine for rendering because it is known for its ability to produce photorealistic results due to its path-tracing algorithm, which simulates the natural behavior of light. This is crucial for achieving realistic thermal effects and shadows in the dataset.

2. **Compute Device Type**

   It is set to CUDA, which enables the use of NVIDIA GPU acceleration. CUDA is particularly effective for speeding up rendering processes by leveraging the parallel computing power of NVIDIA GPUs.
   Device: the rendering device is set to GPU. This choice directs Cycles to use the GPU for rendering rather than the CPU, which results in faster rendering times.

3. **Render Samples**

   The number of samples is set to 640. In cycles, each sample represents a calculation for how light rays accumulate to form the final image. More samples typically mean higher image quality, which leads to less noise and more accurate light representation.

4. **Image Resolution**

   Resolution X: Set to 640 pixels, representing the width of the rendered image.
   Resolution Y: Set to 512 pixels, representing the height of the rendered image.
   These resolution settings determine the size of the output image. A resolution of 640x512 strikes a balance between detail and computational efficiency, making it suitable for datasets where many images need to be processed and analyzed.

5. **Render Output**

   The script dynamically sets the file path for each rendered image, organiz-

ing them into directories based on various parameters such as the angle and background settings.

Figure 2.11: First rendered image with this image background at 40 degrees

Figure 2.12: Second rendered image with this image background at 40 degrees

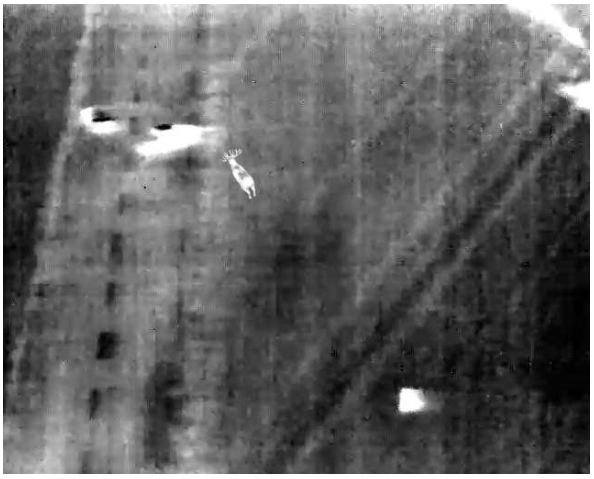

Figure 2.13: First rendered image with this image background at 60 degrees

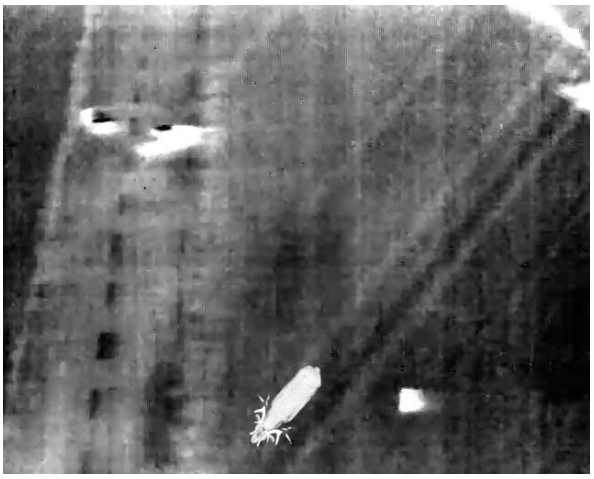

Figure 2.14: Second rendered image with this image background at 60 degrees

2.4 Labels

For each rendered image, there is one txt file containing label information on the axis-aligned bounding box, and the other txt file containing label information on the oriented bounding box for the detected model. The format for an axis-aligned bounding box considers the first class number which represents the detected object, then the x-axis and y-axis coordinates of the object's bounding box, and lastly the calculated width and height of the bounding box. However, the oriented bounding box label file begins the same with the class number but then continues with the coordinates in the following order $x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4$to represent the point-based bounding box shown in the picture below.

x_1, y_1: this is the coordinate of the first vertex of the bounding box, starting from the top-left corner and moving clockwise. This vertex serves as a starting point to define the extent of the box.

x_2, y_2: this coordinate is the second vertex of the bounding box, located in the top-right corner. It defines the horizontal extent of the box from the first vertex.

x_3, y_3: this is the third vertex, found in the bottom-right corner of the bounding box. It indicates the vertical and horizontal limits of the object as defined from the top corners.

x_4, y_4: the fourth and final vertex, located in the bottom-left corner, completes the bounding box. It connects back to the first vertex, enclosing the object.

A point-based oriented bounding box directly uses the coordinates of the four corners $(x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4)$ to define the bounding box without a specific orientation parameter.

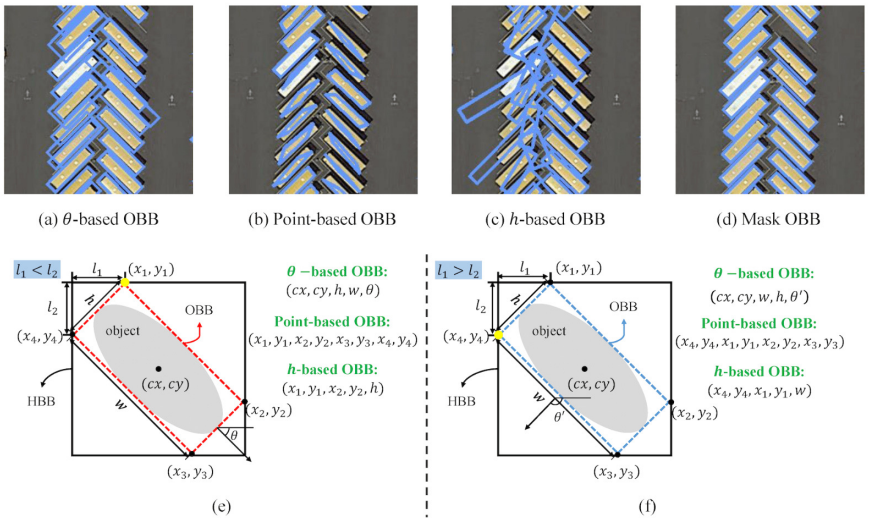

Figure 2.15: Formats for oriented bounding boxes [1]

The final label files ready for training are txt files that contain information about all detected either axis-aligned or oriented bounding boxes. In the folder with axis-aligned bounding boxes, labels are saved in the format of *category_id* x center, y center, width, height. On the other side, in the folder with oriented bounding boxes, labels are saved in format of *category_id*, $x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4$. The first line of the *txt* files is for drone category and its coordinates, which are being calculated in the Scripting part of Blender. Other data of detected bounding boxes is being fetched from the original annotation JSON file with this structure:

```
{
    "info": {
      "year": "string",
      "version": "string",
      "description": "string",
      "contributor": "string",
      "url": "string",
      "date_created": "string"
    },
    "licenses": [
      {
        "url": "string",
        "name": "string"
      }
    ],
    "images": [
      {
        "filename": "string",
        "height": "number",
        "width": "number",
        "id": "number",
        "date_captured": "string"
      }
    ]
}
```


Figure 2.16: The structure of HIT-UAV10 labels for axis-aligned bounding boxes

Every image has its corresponded *txt* label files with all the detected bounding boxes information for both axis-aligned and oriented bounding boxes. The image, its detected axis-aligned bounding boxes *txt* file and its detected oriented bounding boxes all have the same name with right extension.

```
{
    "info": {
        "year": "string",
        "version": "string",
        "description": "string",
        "contributor": "string",
        "url": "string",
        "date_created": "string"
    },
    "licenses": [
        {
            "url": "string",
            "name": "string"
        }
    ],
    "scenarios": [
        {
            "id": "number",
            "name": "string"
        }
    ],
    "weather": [
        {
            "id": "number",
            "name": "string"
        }
    ],
    "images": [
        {
            "id": "number",
            "width": "number",
            "height": "number",
            "depth": "number",
            "filename": "string",
            "scenario": "string",
            "weather": "string",
            "perspective": "string",
            "altitude": "string",
            "date": "string"
        }
    ]
}
```


Figure 2.17: The structure of HIT-UAV10 labels for oriented bounding boxes

Figure 2.18: The structure of MONET labels for axis-aligned bounding boxes

3 Object Detection

3.1 Mesh Detection and Bounding Box

The logic behind object detection and finding its correct bounding box is executed in two functions: `generate_drone_yolo_label` and `generate_2d_yolo_label`. The `generate_2d_yolo_la` function encapsulates a method to calculate the bounding box of a mesh object within the camera space of a scene rendered in Blender. This function is crucial for generating labels, particularly for models like YOLO (You Only Look Once), which require precise annotations in terms of bounding boxes. The function is fundamentally designed to transform geometric data from 3D space into 2D image space, accommodating various camera settings and projection types.
This is an explanation of how this function operates and the principles behind it:

1. **Transformation Matrix Calculation**

   Inverse Matrix - the function starts by computing the inverse transformation matrix of the camera (`cam_ob`). This matrix is crucial as it is used to transform coordinates from the world space into the camera space, reversing any transformations applied to the camera (such as rotations or translations). After this step, the object's vertices can be accurately projected from world coordinates to camera coordinates.

2. **Mesh Transformation**

   Mesh Evaluation - the mesh object (`me_ob`) is evaluated to account for any deformations or modifiers that might affect its geometry at render time.
   Coordinate Transformation - the vertices of the evaluated mesh are then transformed first by the object's world transformation matrix to bring them into the

global coordinate system, and subsequently by the camera's inverse matrix to align them within the camera's coordinate system.

3. **Camera Frame Calculation**

   View Frame - the function retrieves the camera's view frame, which represents the boundaries of the camera's view in its local space. This frame is then adjusted for the camera's perspective.
   Perspective check - if the camera is set to perspective mode, the function calculates a projected frame by adjusting each vertex's coordinates relative to their depth (z value). If a vertex of a detected object is behind the camera, where z is lower or equal to zero, making it invisible to us, it is ignored.

4. **Projection to Image Space**

   Normalization - each vertex's x and y coordinates are normalized to a [0,1] range relative to the camera's view, based on the minimum and maximum x and y values of the camera frame.
   Y Coordinate Adjustment - the y-coordinates are adjusted (1-y) to align with image coordinate systems, where the origin is in the top left corner.

5. **Bounding Box Calculation**

   Oriented Bounding Box (OBB) - if the obb flag is true, the function calculates the oriented bounding box using a method (calculateobb) that fits a minimum area rectangle around the object. This is particularly useful for objects rotated or irregularly shaped within the camera view.
   Axis-Aligned Bounding Box (AABB) - if not calculating an OBB, the function calculates the axis-aligned bounding box by determining the minimum and maximum x and y coordinates from the list of projected points, providing the top-left corner and dimensions of the bounding box.

6. **Output**

   The function returns the calculated bounding box coordinates along with the type of bounding box (OBB or AABB). When calculating an OBB, additional

details such as vertices of the bounding box and points of the convex hull are also returned.

The `calculate_obb` function computes the oriented bounding box (OBB) for a given set of points. This function defines the minimum outline rectangle that can contain a set of points with the least area, oriented along the principal axis of the data. The use of Principal Component Analysis (PCA) and convex hull in this function ensures that the OBB aligns with the data's main orientation, providing an efficient and accurate strategy for the bounding box.
Here is a description of the function's operation:

1. **Principal Component Analysis (PCA) Initialization**

   PCA Setup - the function begins by initializing a PCA object with two components, corresponding to the two principal axes in the 2D space of the input points. PCA is used here to identify the directions (principal components) along which the variation in the data is maximized.
   Data Transformation - the points are then transformed using PCA.fittransform(), which aligns them along these newly found principal axes. This step effectively rotates the data to align with these axes, simplifying the process of finding the minimum bounding rectangle.

2. **Convex Hull Calculation**

   Hull Computation - by using the *ConvexHull* method from the *scipy* spatial module, the function computes the convex hull of the transformed points. The convex hull is the smallest shape that encloses all the given points.
   Vertices Extraction extracts the vertices of the convex hull, which are the subset of the transformed points that form this hull's outer boundary.

3. **Bounding Rectangle Calculation**

   Axis-Aligned Bounding Box in Transformed Space - once the points are aligned along the principal axes and the convex hull is determined, finding the minimum bounding rectangle is straightforward. The function calculates the minimum and maximum x and y values among the vertices of the hull.

Rectangle Corners - these minimum and maximum values define the corners of an axis-aligned rectangle in the PCA-transformed space.

4. **Inverse Transformation**

   By using pca.inversetransform() the corners of the rectangle in the transformed space are then mapped back to the original coordinate space. This step effectively rotates the rectangle back to align with the original orientation of the data.
   Output Preparation - the resulting corners (obbcorners) represent the oriented bounding box in the original coordinate system, and they are returned flattened for ease of further processing.

5. **Error Handling**

   Handling Colinear Points - the function includes an exception handler for QhullError, which may occur if the input points are colinear (i.e., all points lie perfectly on a line) or almost colinear, which can also affect the calculation of the convex hull. In such cases, the function handles the error and returns None for the code to continue to the next point.

6. **Output**

   The function returns two outputs - the corners of the oriented bounding box and the vertices used from the convex hull. The convex hull of a set of points in a plane (or higher-dimensional space) is the smallest convex polygon (or polyhedron in higher dimensions) that can enclose all the points. To simplify it - it is analogous to stretching a rubber band around the outermost points in a set; the shape that the rubber band takes is the convex hull. This concept generally has various applications in different fields such as computer graphics, pattern recognition, geographic information systems (GIS), and machine learning. In context of the function `calculate_obb`, the convex hull is used to determine the smallest area that encloses all vertices of an object after they have been transformed through PCA and projected onto a plane. The convex hull is the tightest fitting boundary around the points, which helps accurately

compute the minimum bounding rectangle called the Oriented Bounding Box (OBB) without encapsulating any additional space. By computing the convex hull, the function effectively simplifies the representation of the object's shape in 2D space. This is crucial when you want to minimize the complexity of the shape for further calculations, such as finding the OBB or other forms of bounding geometries. Using the convex hull reduces the number of points that need to be considered for creating bounding boxes. Only the points on the hull are needed for further calculations, rather than all points in the dataset. This can significantly speed up the process of finding the minimum bounding rectangle, especially in cases where there are many points to consider, without affecting the accuracy of the found bounding box. The convex hull ensures that the bounding box encompasses all points of the object, without unnecessary padding that might occur if the bounding box was calculated from all individual points without considering their convex boundary. This accuracy is mainly important in applications where precise object localization and dimension measurement are critical, like in this case of training data for object detection algorithms. The convex hull provides a robust basis for mathematical operations like PCA transformations applied to determine the OBB. It ensures that the transformations and subsequent operations (like scaling or rotation) reflect the true outer boundaries of the point set.

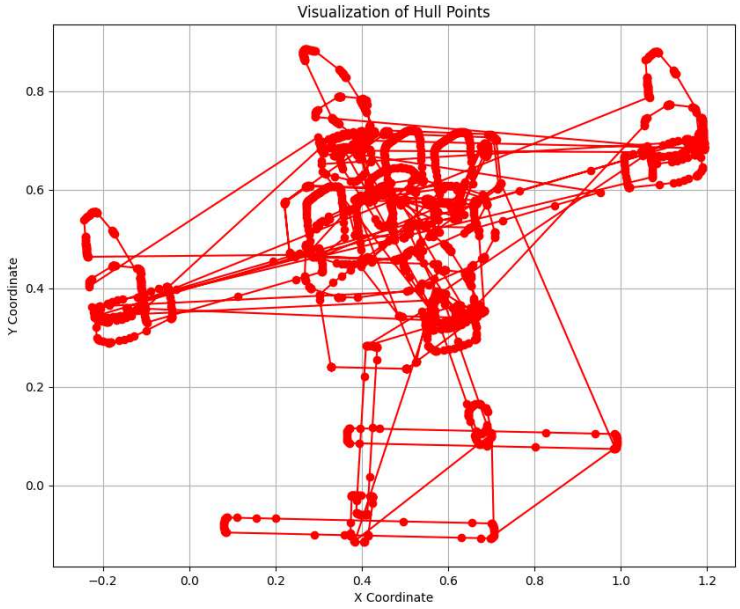

Figure 3.1: Convex Hull visualization

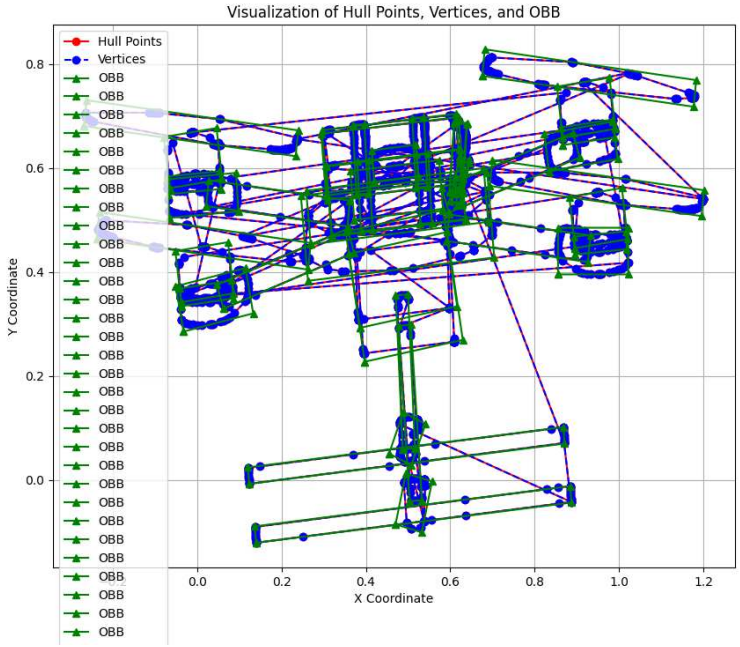

Figure 3.2: Convex Hull, vertices and each mesh's OBB visualization

The function `generate_drone_yolo_label` is designed to create annotations for objects in a collection within a Blender scene, formatted for training YOLO (You Only Look Once) object detection models. The function comprehensively handles both Oriented Bounding Boxes (OBB) and Axis-Aligned Bounding Boxes (AABB) depending on the in-

put flag. The ability to handle both OBB and AABB allows for flexibility depending on the object shapes and orientations in the images, which can significantly affect detection performance in real-world applications.
Here is an analysis of how the function operates:

- **Initialization and Preparations**

  Candidate Lists - lists for x and y coordinates are initialized to store bounding box corners or extents, depending on the bounding box type.
  Collection Retrieval - the function retrieves the specified collection of objects to be labeled. If the collection does not exist, the function exists with an error message.

- **Bounding Box Calculation**

  Object Iteration - the function iterates over objects in the collection. It continues only with mesh-type objects suitable for generating bounding boxes from which 3D models are made.
  Bounding Box Generation - utilizes a helper function (`generate_2d_yolo_label`) to compute the bounding box for each object. This function adjusts for camera perspective, accounting for both perspective and orthographic projections, and considers only objects in front of the camera.
  For oriented bounding boxes, coordinates are stored continuously, as they require different handling than AABB. However, for axis-aligned bounding boxes, corner coordinates are directly used to determine the minimum and maximum extents.

- **Bounding Box Aggregation (for AABB)**

  If not using OBB, the function calculates a combined bounding box for all objects in the collection by calculating the center of the bounding box and its dimensions. It determines the center of the bounding box based on the minimum and maximum x and y values, then the width and height of these extremities.

- **Oriented Bounding Box Calculation (for OBB)**

If OBB is enabled, after collecting all individual object points, an additional OBB calculation is performed using calculate-obb, which involves PCA to determine the minimum area rectangle encompassing all points.

4 Neural Networks Training

In this paper, for real-time object detection, Ultralitics YOLOv8 was used as an image detection model. YOLOv8 is the latest version of YOLO models (as of early 2023) by Ultralitics that builds on the already achieved success of previous versions with new features and improvements for better performance, flexibility, and efficiency. The models represented a few big changes, such as anchor-free detection, the introduction of C3 convolutions, and mosaic augmentation. It is also so-called state-of-the-art model with an outstanding performance. The name itself refers to the model's ability to predict every object present in an image with one forward pass. With YOLO models, the object detection task was reframed as a regression problem which is predicting the bounding box coordinates, not a classification problem as it was used to. YOLO models are pretrained on a huge amount of datasets of different formats like COCO, ImageNet, PASCAL VOC... This is why they provide highly accurate predictions on classes that they have been pretrained on - that is their master ability. In addition, the models can also learn new classes comparatively easily - that is their student ability. Finally, that is why it is said that the models have the coeval ability to be both the Master and the Student. Users are authorized to share, modify and distribute the software for free.

There are two main differences in the new YOLOv8 modes and YOLOv5, its predecessor - Anchor-Free Detection and Mosaic Augmentation. Anchor-Free Detection is based on anchor boxes. Before them, an object is assigned to a grid cell containing the given object's midpoint. The problem with making bounding boxes and allocating them would appear if two objects had the same center point.

In this situation, both objects have the same center point. Now is when the usage of anchor boxes is very effective. They are something like cookie-cutter templates. Two anchor boxes are created and then the one that has the highest overlap with the ground

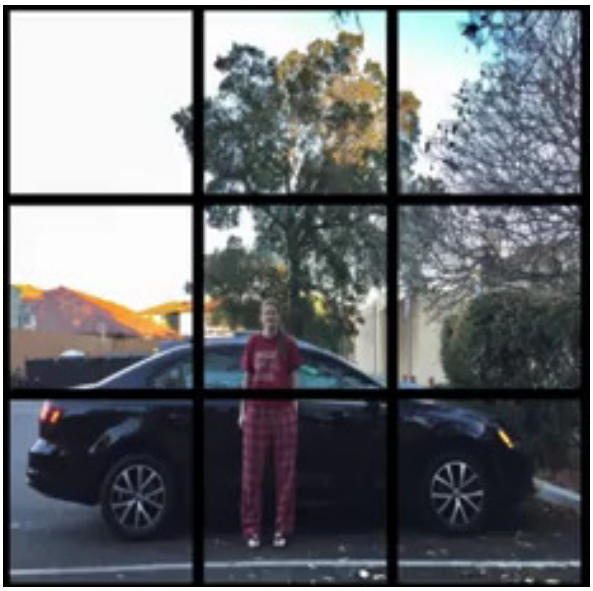

Figure 4.1: Objects with the same center point

truth bounding box and assign that box to the corresponded class. They are very practical because they can be both horizontally and vertically elongated like detected figures. By using Anchor Boxes, the training was improved as increasing MAP. Anchor Boxes were then incorporated in previous YOLO models. However, there are two reasons why aren't they used in YOLOv8 model - lack of generalization and lack of proper anchor boxes in irregularity. If there is any lack of generalization, it means that the final model is hard to fit on a new data and is overtrained, which is not a good thing. With polygon anchor boxes, irregularities can't be mapped.

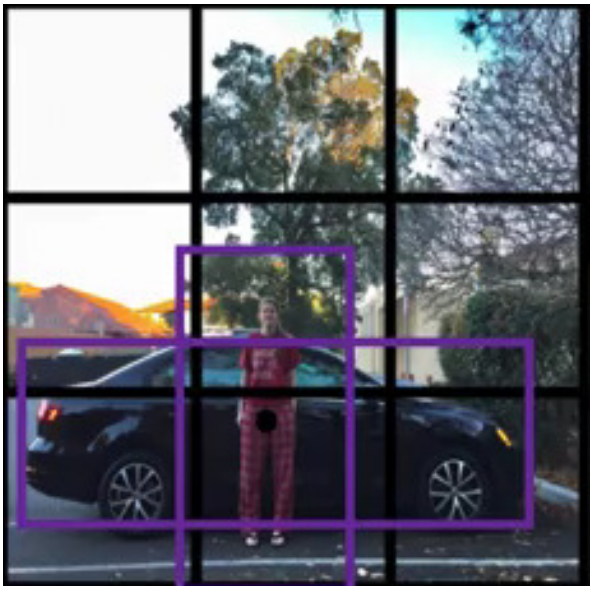

Figure 4.2: Objects' anchor boxes

Mosaic data augmentation is one of the augmentation that YOLOv8 model does to training images. It is a simple technique where four different images are stitched together creating an input for the model to be feed with. The main reason is to enable the model to learn the actual objects from different positions and in partial occlusion.

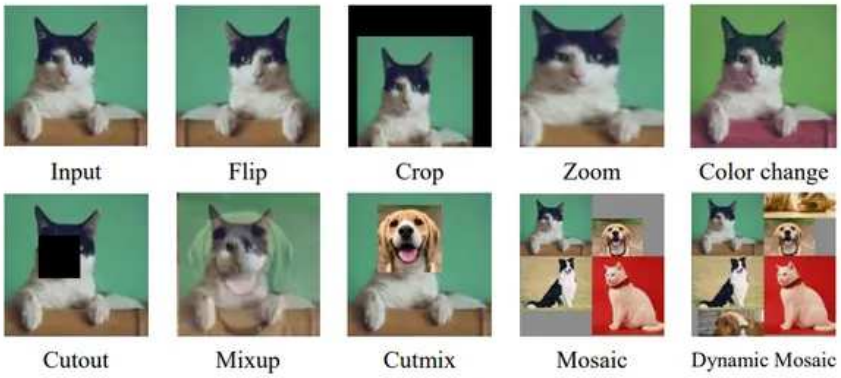

Figure 4.3: Mosaic data augmentation example

The inference time is much faster than all the other YOLO models. YOLOv8 model gives around 33(percent) more MAP for n-size models and greater MAP across the board. In general, the model size is linearly proportional to MAP and inversely proportional to interference time. That is why smaller models have faster inference time but have a lot lesser mAP. They are also more efficient if we have less space, meaning having less edge scenarios. On the other side, bigger models are better if we have less data.

The HIT-UAV10 dataset underwent two distinct training phases—initially for axis-aligned bounding boxes and subsequently for oriented bounding boxes. Identical procedures were employed during both training sessions. Both sets of labels were split into train, validation and test folder at an 80-10-10 ratio. The only difference in training was the loaded model. For training labels of axis-aligned bounding boxes the *yolov8s.pt* is used, while for training labels of oriented bounding boxes the *yolov8n-obb.pt* was used. These are the training parameters: the learning rate of the algorithm is set to 0.001. It controls how much to change the model in response to the estimated error each time the model weights are updated. It affects how quickly or slowly a neural network learns. Current epochs are set to 3000 meaning the learning algorithm works through the entire dataset 3000 times. In general, an epoch is one complete presentation of the dataset to be learned to a learning machine. The current batch size is a number of training examples utilized in one iteration, and it is set to 64 meaning the 64 samples from the training dataset will be used to estimate the error gradient before the model weights are updated. It is somewhat a balance between the computational efficiency of larger batches on one size and the robustness of smaller batches effects on the other. Finally, current image size is set to 640 as this is the size to which all input images are resized before being fed into the model.

The MONET dataset was trained for detecting only oriented bounding boxes on the thermal model of deer in order to see the applicability of oriented detection to everyday problems in agriculture and forestry. Since original labels don't have information on the angle of the object's bounding box, the training was done only on the deer. Again, set was split into train, validation and test set at a ratio 80-10-10. The model *yolov8n-obb.pt* was used as well as all the other parameters as in the case of training HIT-UAV10 dataset for oriented bounding boxes.

After training, test data was tested in a way that the bounding boxes were visualized on the test part of the images to see the succession of the model.

5 The Results

5.1 Training HIT-UAV10 dataset on axis-aligned bounding boxes

Looking at the normalized confusion matrix, it is noticeable that there are both strengths and weaknesses of a model. With classes like person, car, and bicycle, the model performs very well. This is initiated by high numbers on the main diagonal for these classes, showing that the model can reliably identify those objects. In addition, the model effectively identifies background with relatively few misclassifications compared to other classes. This is indicating that the model is very well-tuned to distinguish object areas from non-object areas in images. However, the model shows some confusion between classes that are potentially similar in appearance or contextually close in the image. That can be seen for classes car and other vehicle, as well as for the bicycle and other vehicle. The drone shows perfect classification in this context, which is excellent. The model overall shows strong performance across most classes, but could potentially benefit from improvements in distinguishing similar objects and reducing misclassifications involving the background.

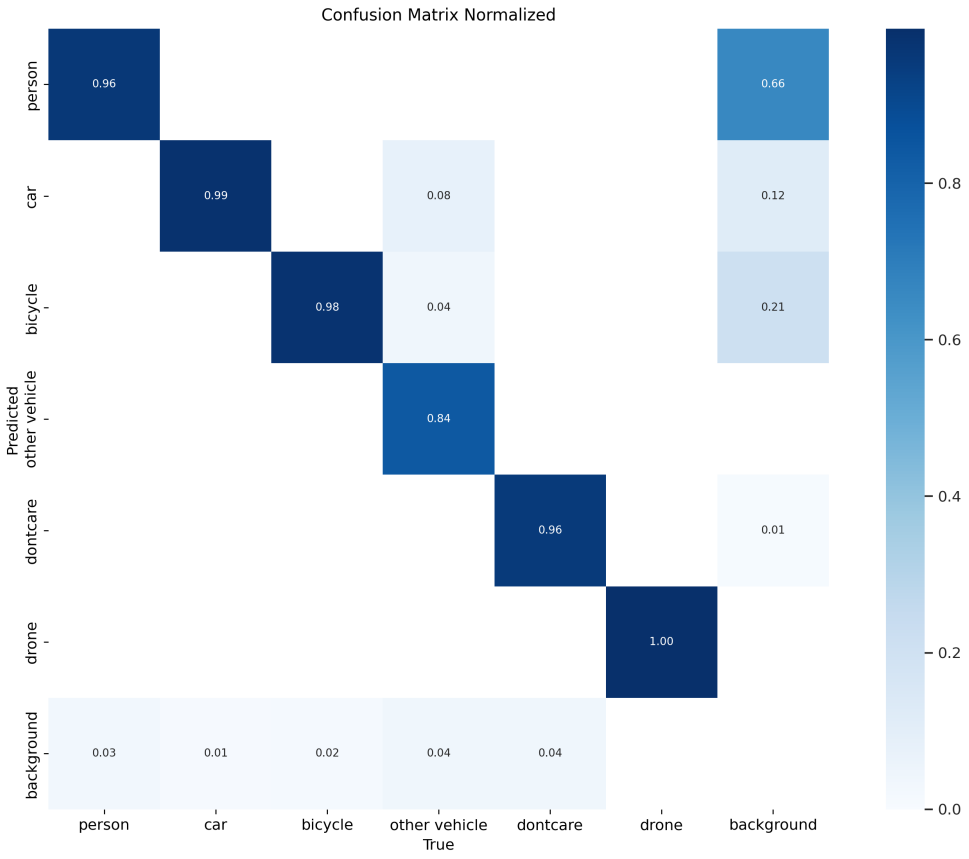

Figure 5.1: Normalized confusion matrix

As shown, the box loss for both train and validation set show the loss associated with bounding box prediction, meaning how off are the predicted bounding boxes from the true ones. A lower value represents that the model is getting better at accurately predicting the location of objects. The training loss decreases consistently, indicating learning, while the validation loss shows some fluctuation, which is typical as the model tries to generalize from training to unseen data. The cls loss graphs depict the classification loss during training and validation. What the loss measures is how well the model is performing in correctly classifying the objects within the predicted bounding boxes. The visible downward trend in the training graph is good. However, there are some fluctuations in the validation graph, suggesting variability in how well the model generalizes its classification ability to new data. The final set of graphs dfl loss are related to how well the model predicts certain attributes or orientations of the objects. Good learning and adaptation are visible from both graphs, even though validation loss again shows greater variability. Both precision and recall are critical for evaluating the effectiveness of the mode. Precision measures the ratio of correct positive observations to total pre-

dicted positives, while recall measures the ratio of correct positive observations to all observations in actual class B. Both metrics are generally high, suggesting good model performance. mAP50 calculates the average precision at 50 percent IoU threshold, while mAP50-95 calculates averages over a range from 50 to 95 percent IoU. The metrics show how well the model is detecting objects according to different criteria for overlap between the predicted bounding boxes and the true ones. High values across these metrics indicate robust detection capabilities.

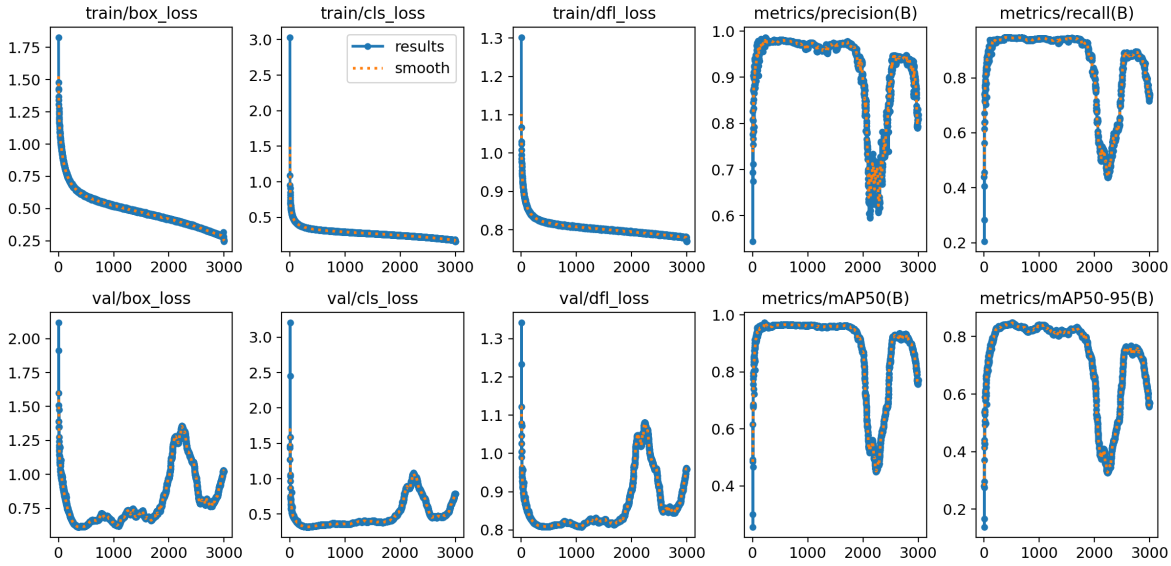

Figure 5.2: Results of the training

When discussing all classes combined case, the combined F1 score peaks around 0.96 at a confidence threshold of 0.619 indicating excellent overall performance. The model generalizes well across multiple classes, maintaining high F1 scores, particularly for person and car classes, which suggests strong feature learning and classification capabilities.

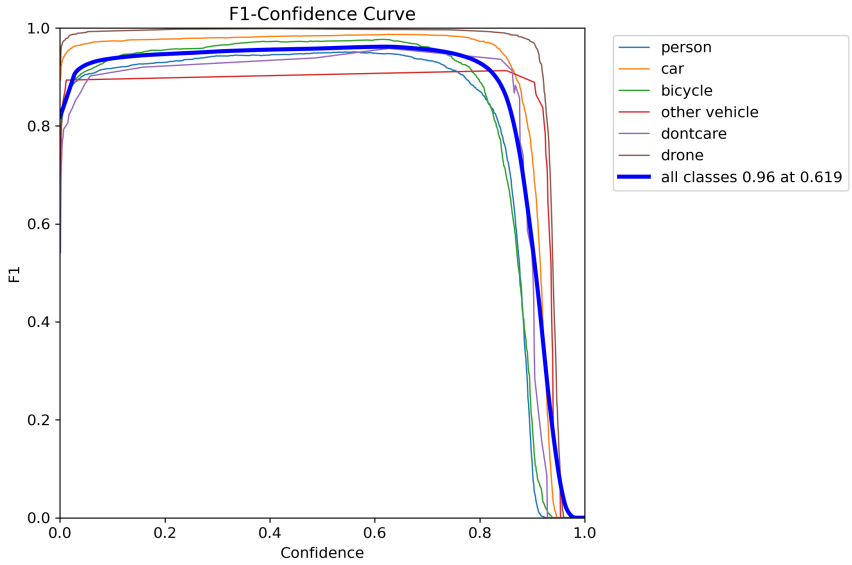

Figure 5.3: F1 score curve

Overall PR curve is very close to the top-right corner, which indicates a better performing model with high precision and high recall. The blue line represents an average precision across all classes at a 50 percent IoU threshold. In general, a mAP of 0.965 is exceptionally high. When looking at individual classes, the drone shows the highest precision across almost all recall levels, which indicates exceptional accuracy in predicting this class.

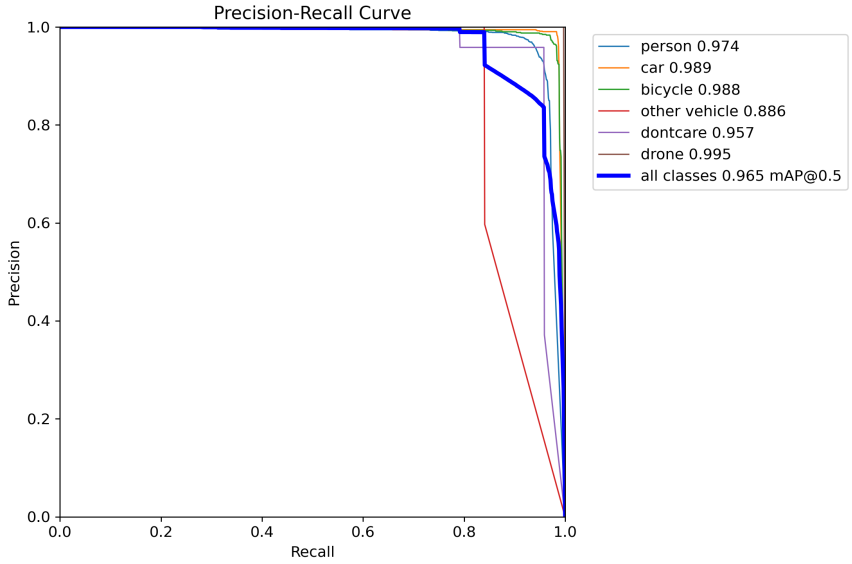

Figure 5.4: Precision-recall curve

The graph shows that precision increases across all classes as the confidence threshold increases. This is also expected, since higher confidence in most cases correlates with more reliable predictions. Here again, drone exhibits nearly perfect precision across almost all confidence levels, particularly at higher thresholds. This means that when a model predicts an instance as a drone, it is almost always correct.

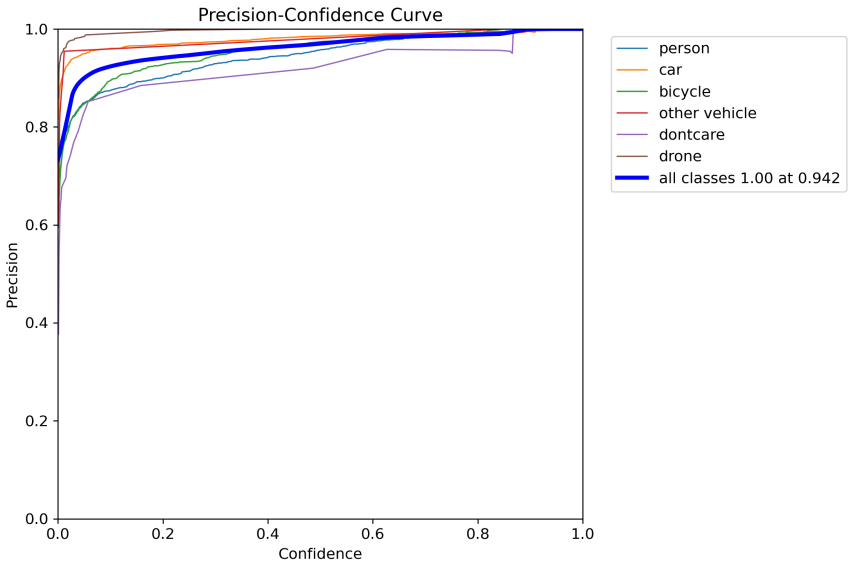

Figure 5.5: Precision curve

As expected, recall decreases as the confidence threshold increases because setting a higher threshold results in fewer positive predictions, making the model more conservative. Again, drone's recall maintains high across a wide range of confidence thresholds, meaning the model is highly effective at detecting almost all true drone instances, even as the threshold increases.

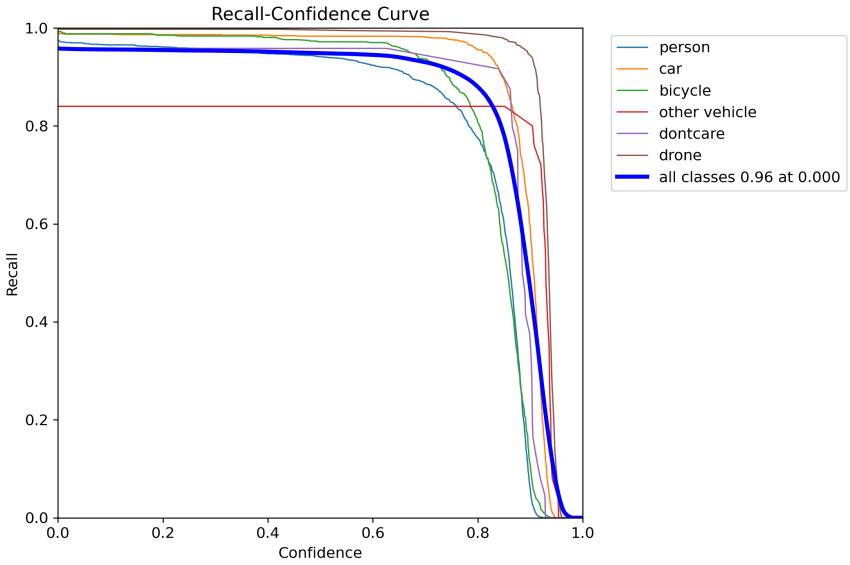

Figure 5.6: Recall curve

Validation Batch Labels images depict the ground truth labels for distinct batches from the validation dataset, providing a clear picture of what the objects are and their respective locations as per the dataset. Validation Batch Predictions visuals display the predictions made by the YOLOv8 model for the respective batches. Comparing these two to the label image, it is visible how well and highly effective the model detects and classifies objects visually.

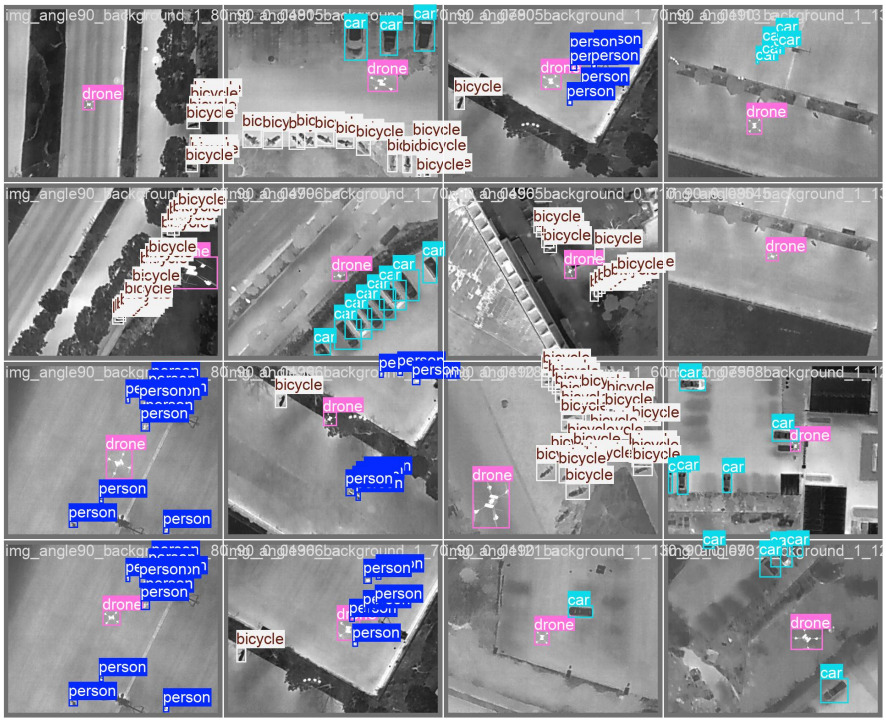

Figure 5.7: Validation batch labels 1

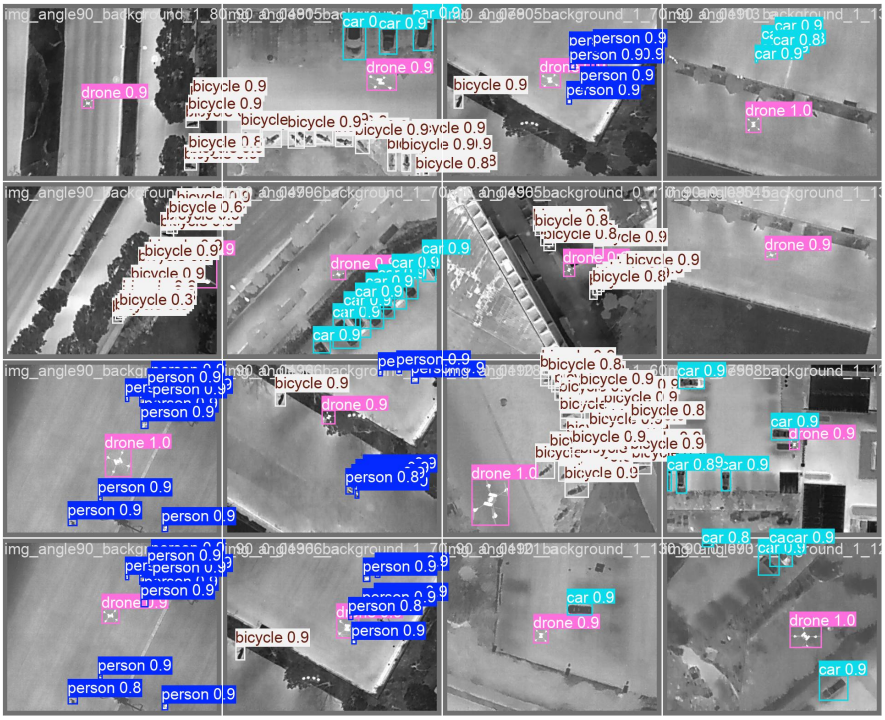

Figure 5.8
Validation batch predictions 1

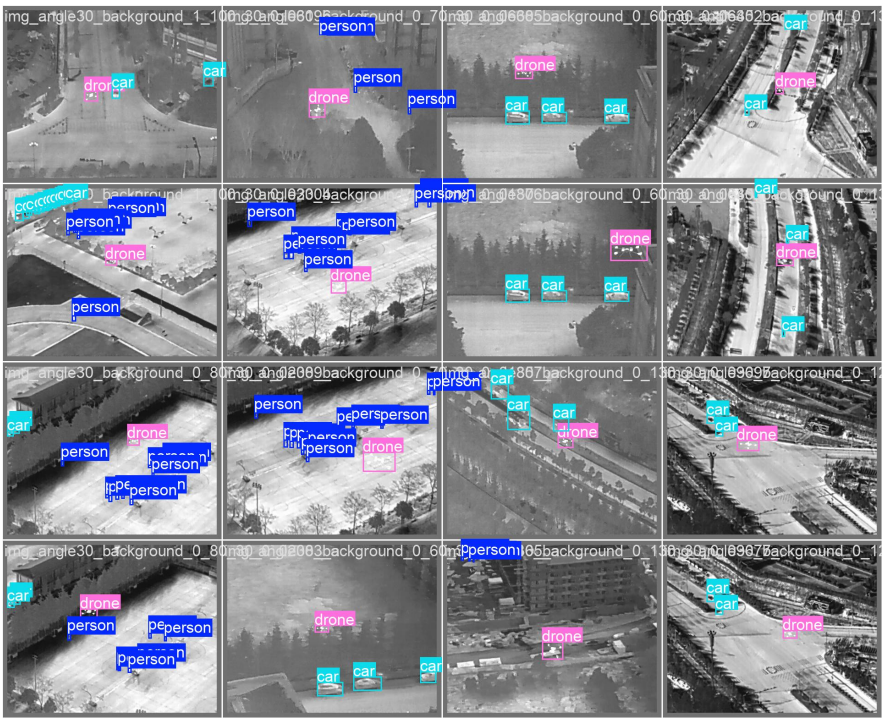

Figure 5.9: Validation batch labels 2

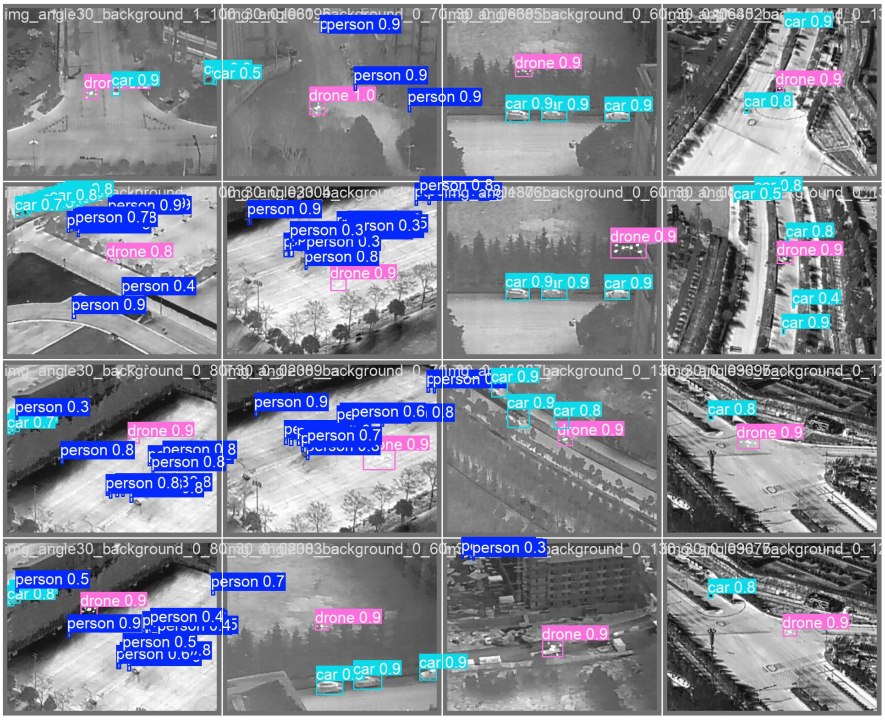

Figure 5.10: Validation batch predictions 2

The final step for axis-aligned labels set was to test the trained prediction on a test part of the set. Here are the detected classes based on the trained model. It can be observed that the detection is very well performed, making this model highly effective.

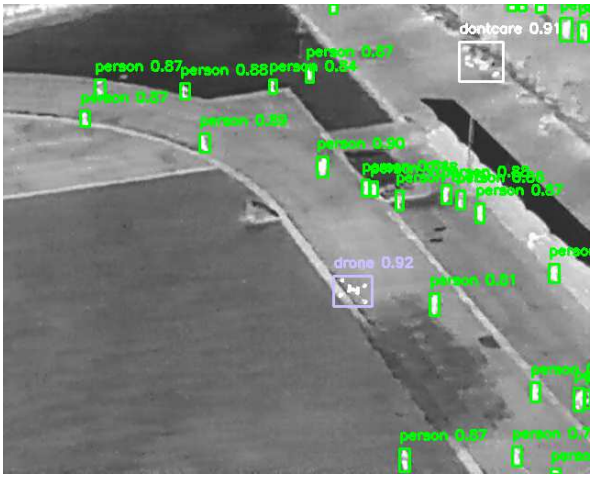

Figure 5.11: Image at 30 degrees with detected objects

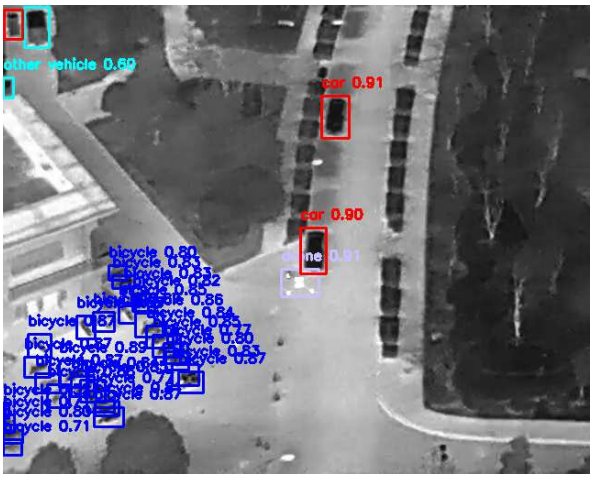

Figure 5.12: Image at 40 degrees with detected objects

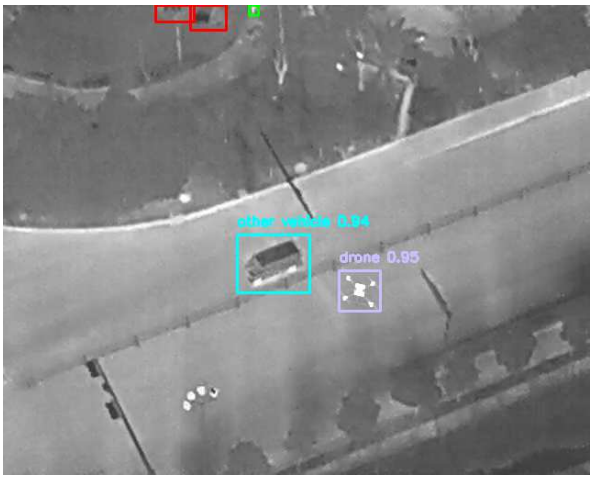

Figure 5.13: Image at 50 degrees with detected objects

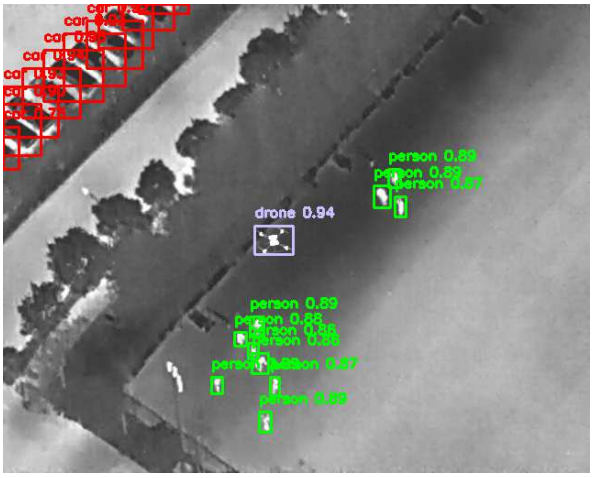

Figure 5.14: Image at 60 degrees with detected objects

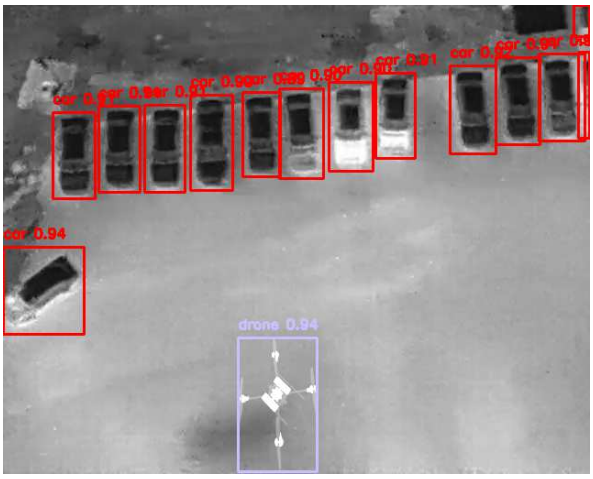

Figure 5.15: Image at 70 degrees with detected objects

Figure 5.16: Image at 80 degrees with detected objects

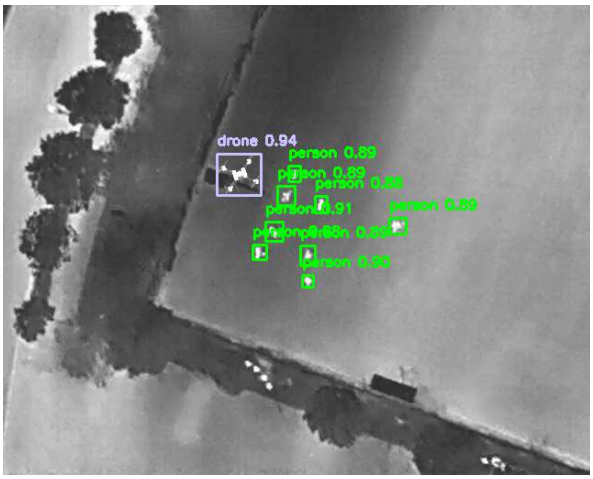

Figure 5.17: Image at 90 degrees with detected objects

5.2 Training HIT-UAV10 dataset on oriented bounding boxes

The high main diagonal values - Person (0.97), Car (0.99), Bicycle (0.96), and Drone (1.00) indicate the high accuracy of the model's classes prediction. There are precise recogni-

tion capabilities for the drone category, since the diagonal value is 1. On the other hand, when talking about misclassifications and observing non-diagonal values, there are indicators that the model has some difficulties with the dontcare category.

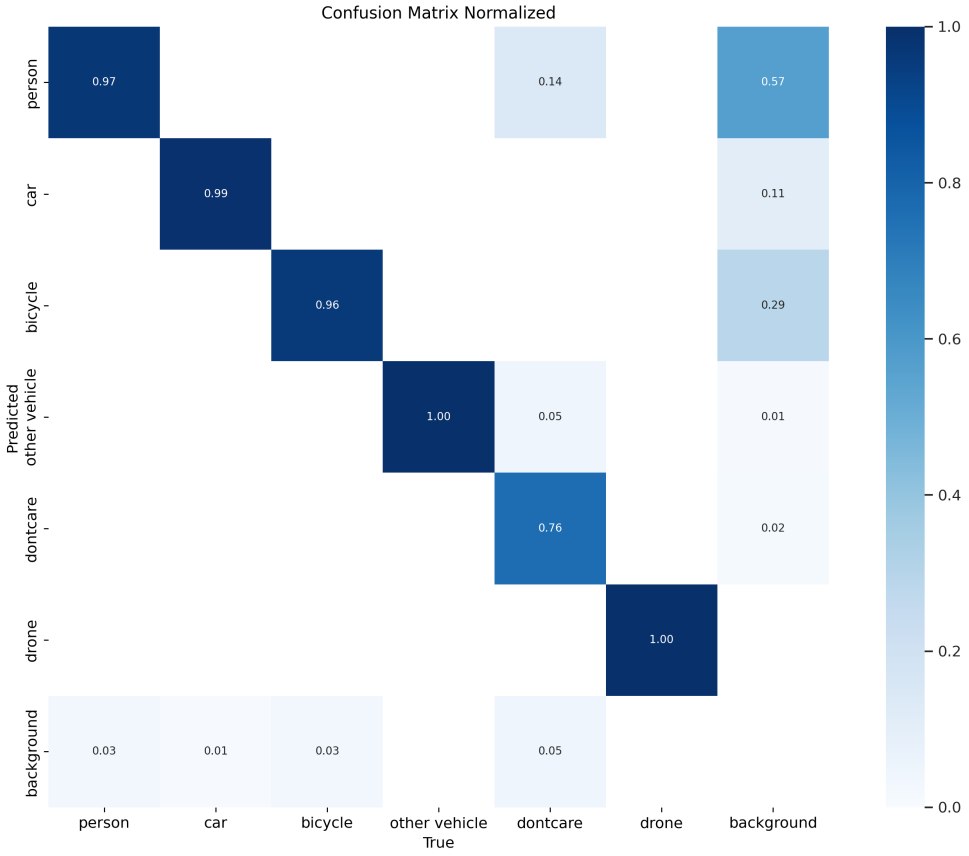

Figure 5.18: Normalized confusion matrix

A decreasing trend, that is shown in both training and validation box losses, indicates that the model is effectively learning to predict the bounding boxes around objects in the images. The smoother line for training and relatively stable validation loss suggest that the model is not overfitting significantly on the training data. Classification loss decreases over time in training, just like box loss. This means improvement in the model's ability to correctly classify the objects within the detected boxes. The validation loss has some fluctuation, as expected in validation phases, due to variability in unseen data. Dfl loss has a consistent decrease in training loss, which is usual. However, there are again some fluctuations in validation loss, but it is still a relatively stable validation loss. The results of precision and recall metrics show high precision and recall at the beginning, which could indicate a model performing well on more apparent features or labels in the dataset. However, both metrics experience a decline as training progresses,

potentially indicating the model's struggle with more complex, varied, or less frequent features as it sees more data. The mAP graphs show high initial scores that decrease over time, particularly noticeable in the mAP50-95 graph. This might suggest that while the model initially does well at detecting objects generally, its ability to precisely match the ground truth bounding boxes with high IoU thresholds diminishes slightly, possibly due to learning more complex data or beginning to overfit to less representative aspects of the training data.

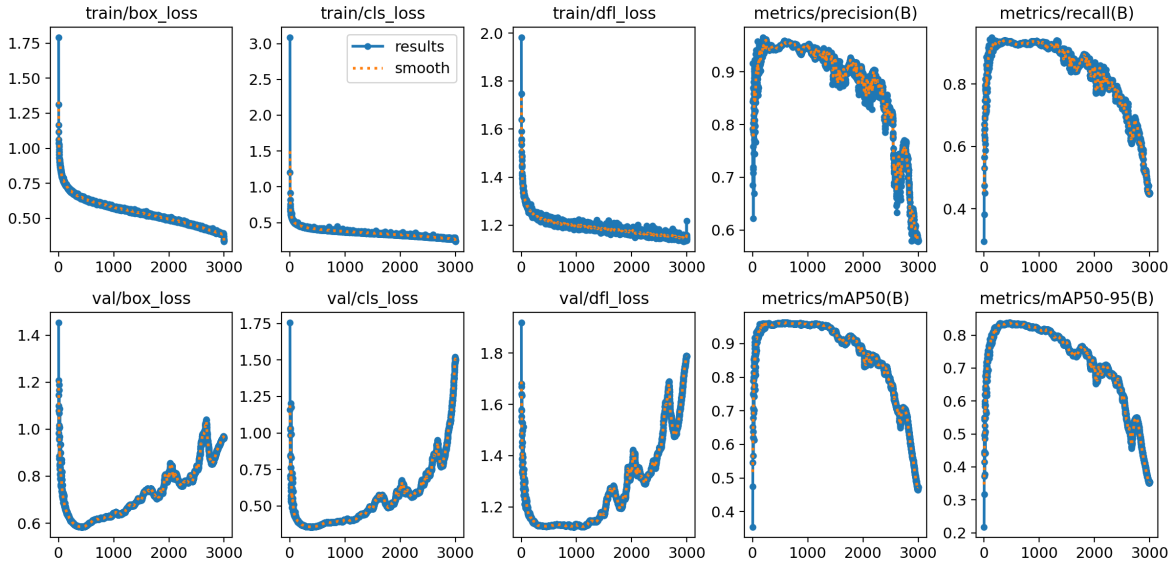

Figure 5.19: Results of the training

For most classes, the F1 score starts high at lower confidence thresholds, which means that the model has good recall and good precision at these levels. As confidence thresholds increase, there is a period where F1 score is at a high level, but then sharply drops off. This can be interpreted in a way that the precision increases with confidence but at the cost of recall. The line representing all classes says that the model achieves an F1 score of 0.94 at a confidence threshold of 0.633 for all classes combined. This shows that the model is generally accurate and balances in predicting at this threshold across all classes. Here, drone exhibits a stable high performance until a sharp drop, suggesting good model performance for this class at lower to moderate confidence levels.

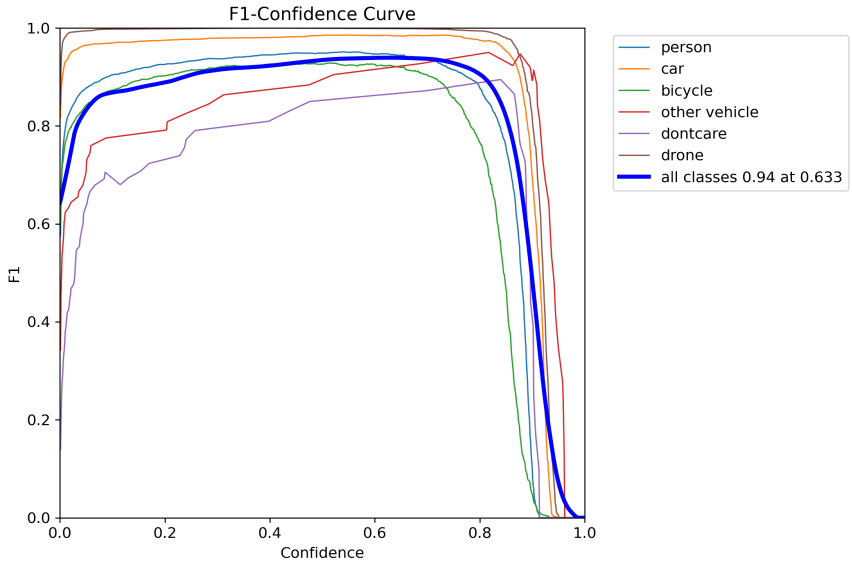

Figure 5.20: F1 score curve

It can be observed that precision increases for all classes as the confidence threshold is raised, as expected due to higher confidence thresholds excluding less certain predictions which are more likely to be false positive. The all classes line represents that at a confidence threshold of 0.928, the model achieves a precision of 1 for all classes combined. This is an excellent indicator of the model's overall precision capability at high confidence levels. Drone shows high precision across a wide range of confidence thresholds, meaning the model has a great ability to detect drones with high reliability.

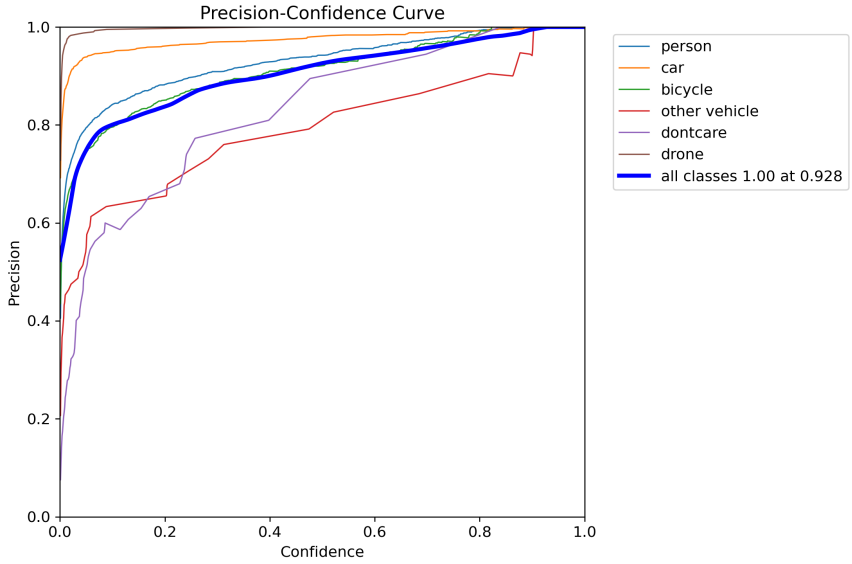

Figure 5.21: Precision curve

The model's overall performance across all classes is summarized, showing an aver-

age precision of 0.960 at an IoU threshold of 0.5. This high score indicates excellent average performance but suggests there is some variation between classes. Again, drone's curve shows almost perfect precision across nearly all levels of recall. This suggests that the model is extremely accurate and reliable in predicting the class of drone. Car class also shows excellent precision, while other vehicle class shows the lowest precision among the classes.

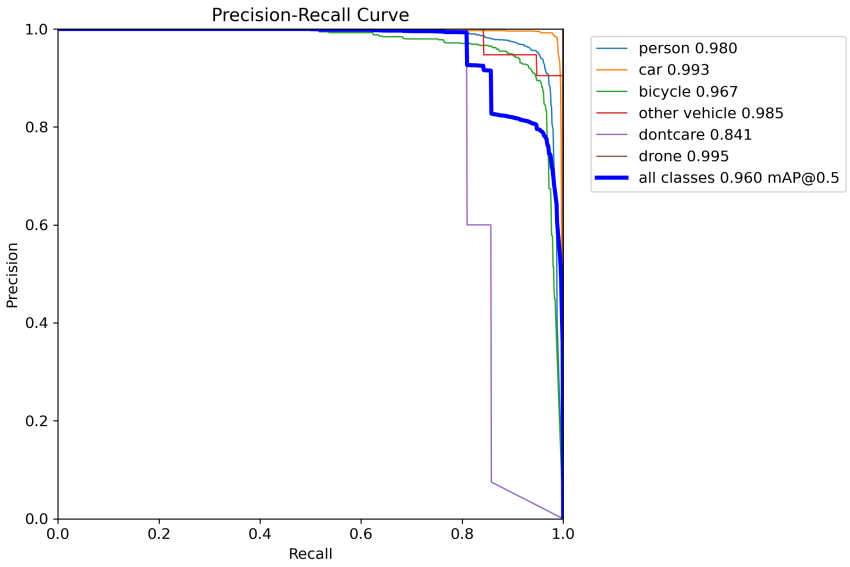

Figure 5.22: Precision-recall curve

The model achieves an overall recall of 0.97 at the lowest confidence threshold (0.00), which indicates excellent initial sensitivity. This high value across all classes suggests that the model is capable of detecting nearly all relevant instances when it is allowed to make predictions freely (with minimal confidence restriction). Drone and car classes maintain higher recall values even as the confidence threshold increases, meaning that the model is very effective at detecting these objects. Dontcare class show the lowest recall across all confidence thresholds, probably due to not precisely enough defined what is considered as a dontcare object.

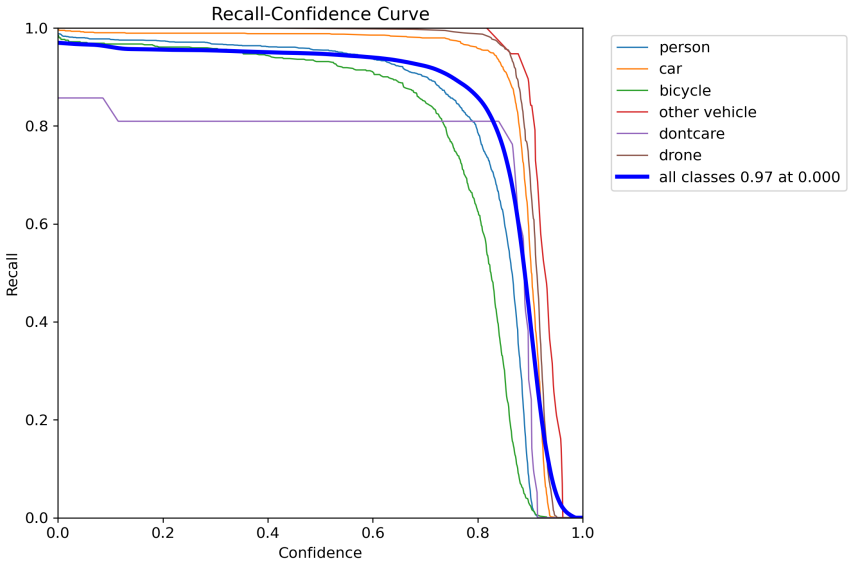

Figure 5.23: Recall curve

When comparing visuals of validation batch predictions and images of validation batch labels, it is immediately visible how the model is excellent at detecting objects and classifying them correctly.

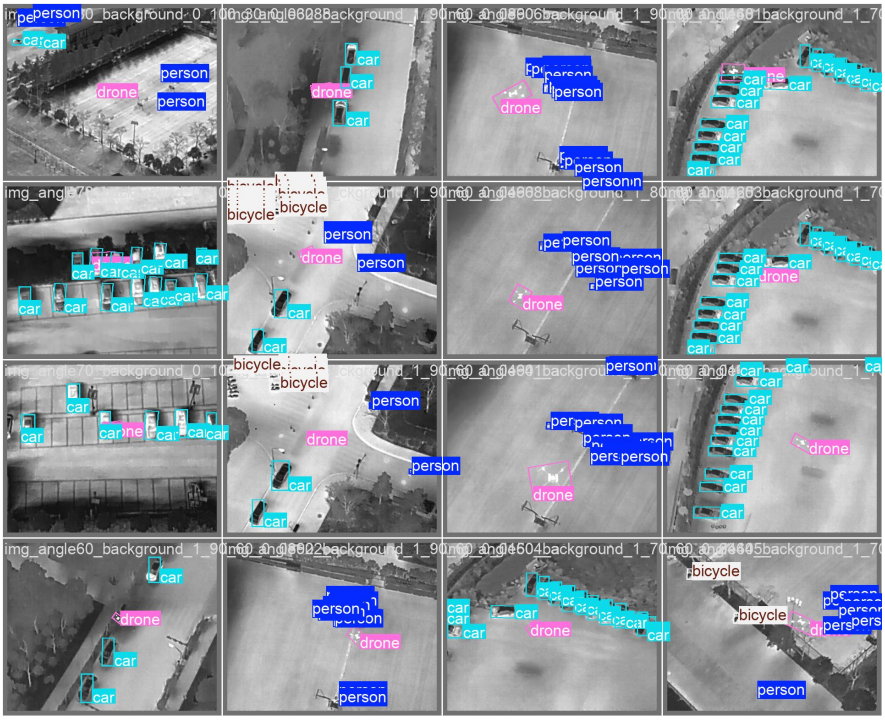

Figure 5.24: Validation batch labels 1

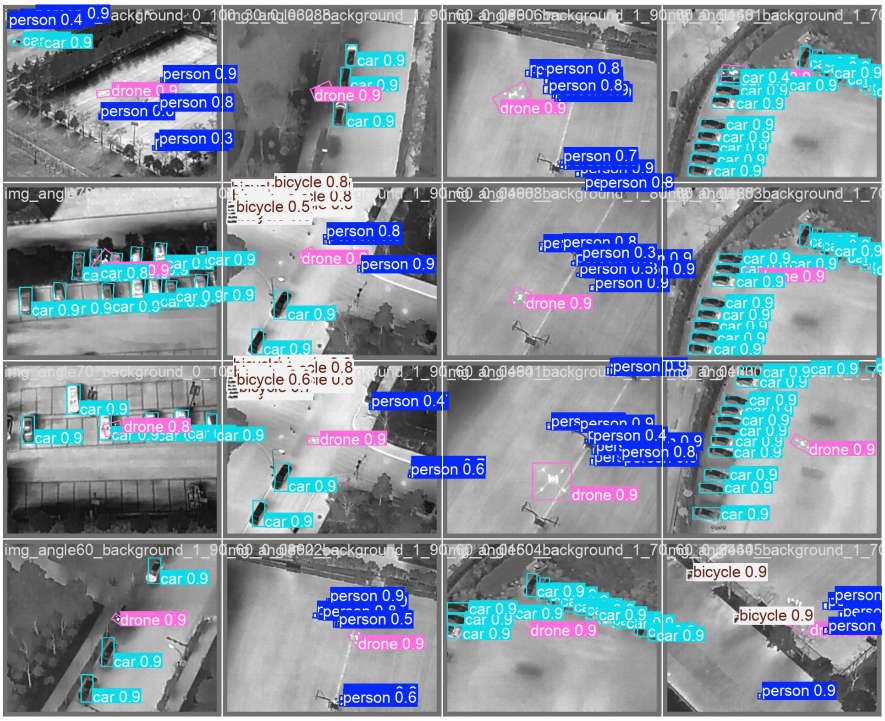

Figure 5.25: Validation batch predictions 1

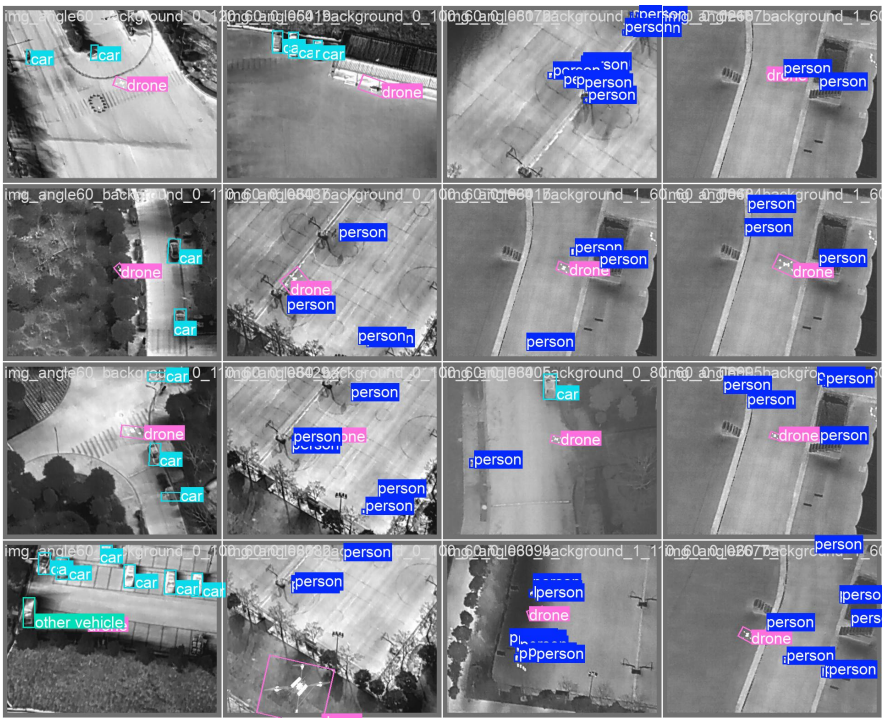

Figure 5.26: Validation batch labels 2

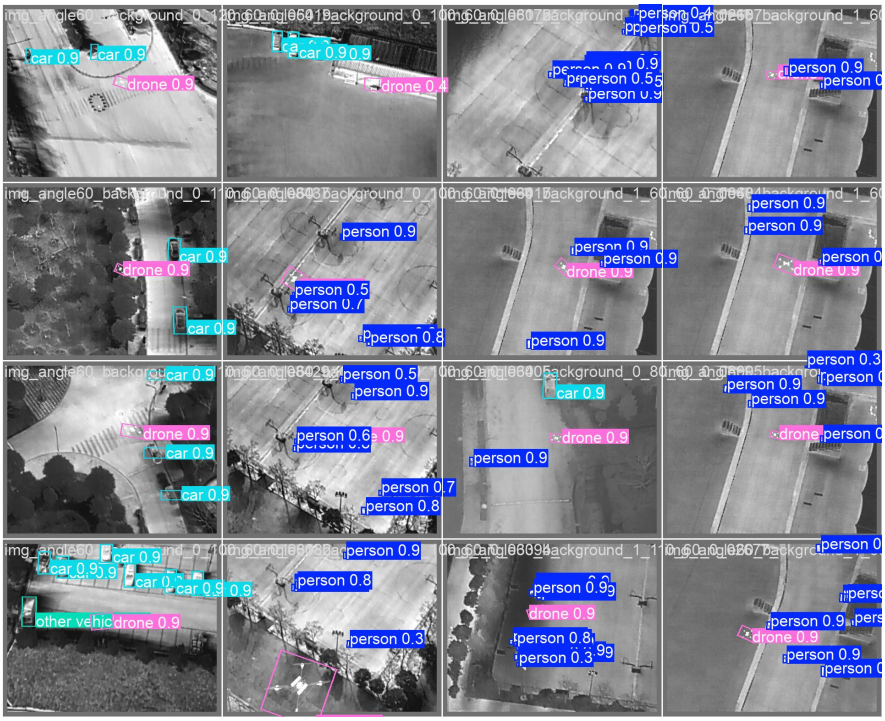

Figure 5.27: Validation batch predictions 2

After the training, the visualization was made on a test set to validate the trained prediction. The detected classes match real classes and bounding boxes of all objects. This shows the accuracy and success of the trained model.

Figure 5.28: Image at 30 degrees with detected objects

Figure 5.29: Image at 40 degrees with detected objects

Figure 5.30: Image at 50 degrees with detected objects

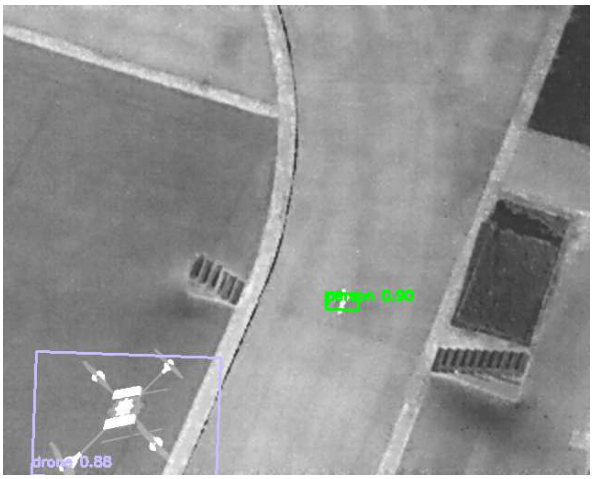

Figure 5.31: Image at 60 degrees with detected objects

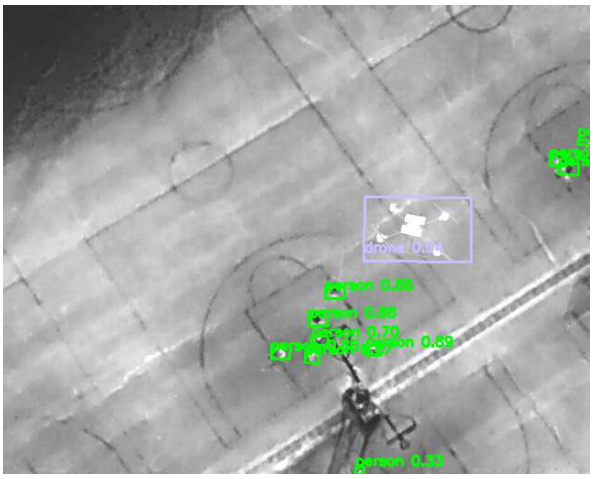

Figure 5.32: Image at 70 degrees with detected objects

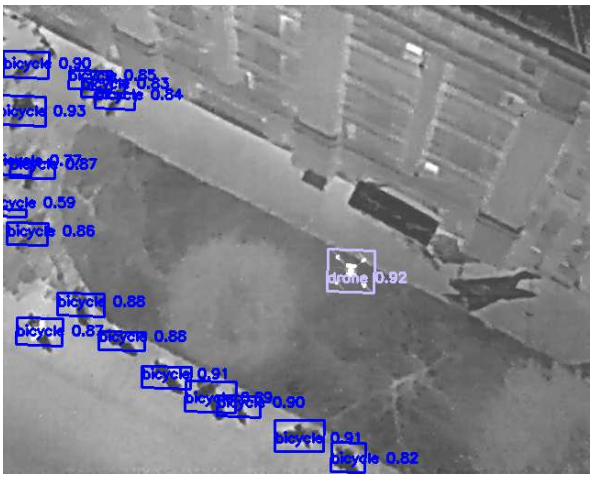

Figure 5.33: Image at 80 degrees with detected objects

Figure 5.34: Image at 90 degrees with detected objects

5.3 Training MONET dataset on oriented bounding boxes

It can be read from the matrix that the model predicts the deer correctly 93percent of the time when it is actually a dear. It is quite effective in identifying that class. However, there are some false negatives when the model misclassifies deer as background class, which happens only 7percent of the time.

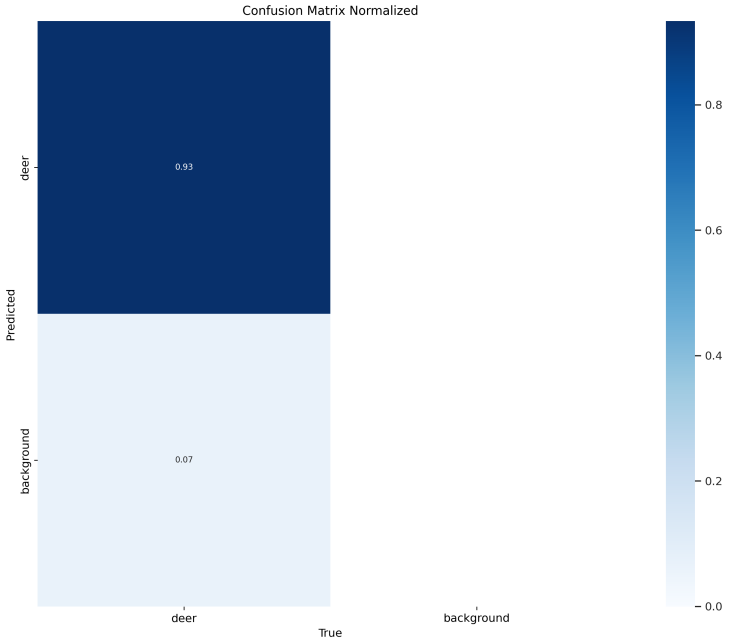

Figure 5.35: Normalized confusion matrix

The graph of box loss shows a sharp decrease initially, which then stabilizes as training progresses, meaning the model quickly learned the appropriate features for bounding box prediction and continued to refine its ability gradually. There are some differences between validation loss, but still the similarity is high and suggests that the model is generalizing well to unseen data without overfitting. Classification loss also shows a step decline, followed by stability. Then this is followed by validation classification. The validation loss is slightly more inconstant, but that is typical in validation phases due to the diversity of unseen data. The Direct Feature Labeling (DFL) loss is related to the accuracy of attribute prediction within the model. It decreases significantly, indicating good learning dynamics. On the other side, the DFL validation loss curve is more variable due to possible challenges in generalizing those attribute prediction to new data. This might indicate areas where the model could benefit from additional training or data augmentation. Precision remains high throughout the training, which indicates a low rate of false positives. Recall shows a steep increase to peak levels and then stabilizes, which means the model is correctly identifying a high percentage of all positive samples in the dataset. At an Intersection over Union (IoU) threshold of 0.5, mAP50 curve shows high levels of Mean Average Precision. From that, it can be concluded that the model localizes and identifies well. The mAP50-95 metric also shows high values in average scores across a range of IoU thresholds from 0.5 to 0.95. This means that the model performs well not

only at a basic level of IoU, but also on more strict criteria.

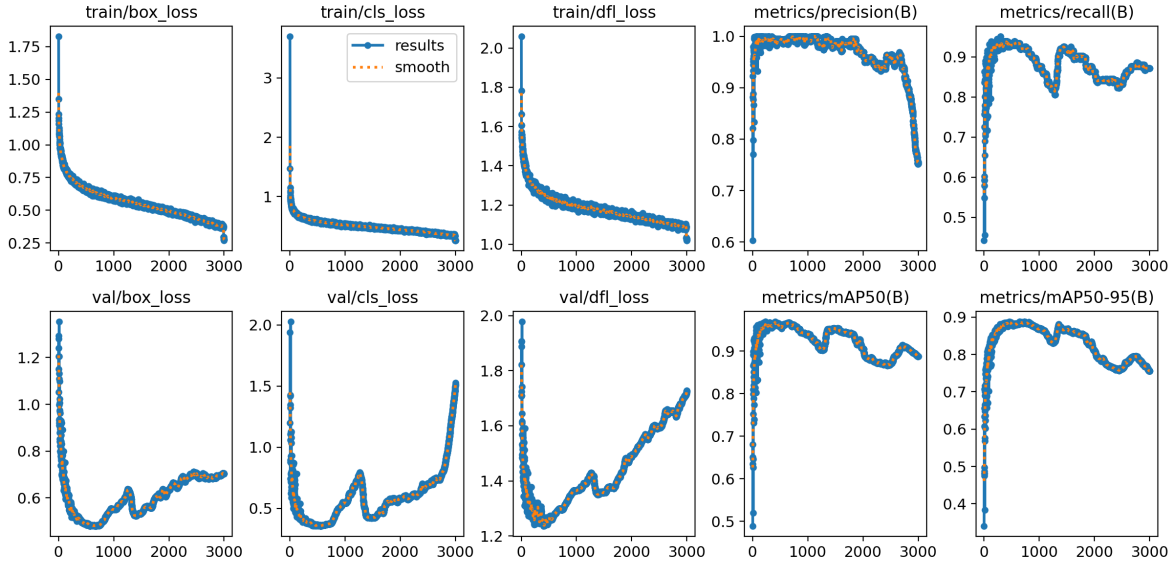

Figure 5.36: Results of training

The F1 score is extremely high (close to 1.0) for most confidence thresholds up to about 0.617, at which it peaks with a score of 0.96. There is a sharp drop in the F1 score as the confidence threshold increases beyond approximately 0.617. This suggests that beyond this threshold, either precision or recall (or both) decreases significantly. The optimal confidence threshold for deployment would be around 0.617, where the F1 score is maximized, balancing precision and recall effectively.

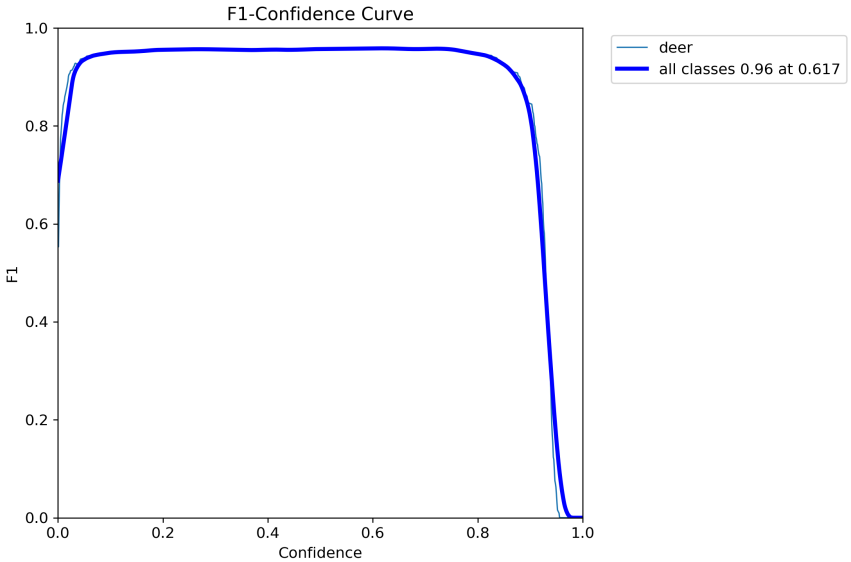

Figure 5.37: F1 score curve

The model is extremely precise when predicting the presence of a deer, with making very few false positive errors, which can all be seen from the fact that the Precision curve starts very high near 1.0 even at low confidence thresholds. At a confidence threshold of 0.756, the model achieves perfect precision across all classes. This is an optimal point where every prediction the model makes across all classes is correct, albeit potentially at the cost of missing some true positives (lower recall).

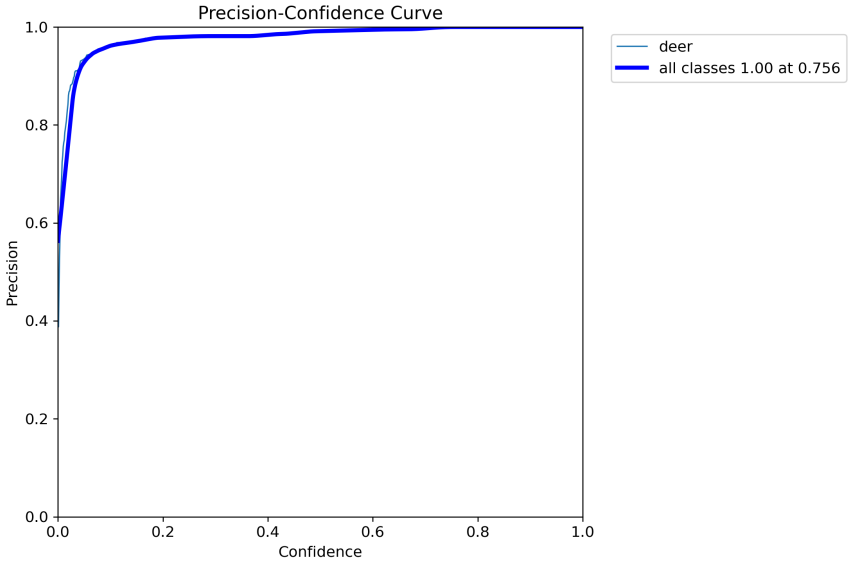

Figure 5.38: Precision curve

The Recall curve also starts with a very high precision near 1.0 for low recall values and maintains high precision as recall increases. Because of that, the model can identify true positives while maintaining a low rate of false positives.

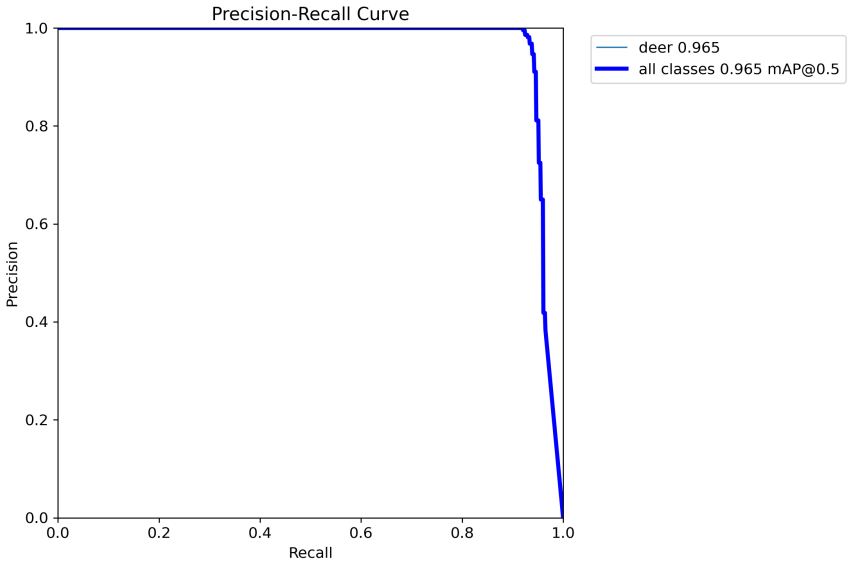

Figure 5.39: Precision-recall curve

The behavior of having a sharp drop in precision as the recall approaches the value of 1.0 is expected because in order to achieve nearly complete detection (high recall), the model starts to make more false positive errors. The drop-off point suggests the threshold beyond which the model begins to sacrifice precision to detect more positive samples.

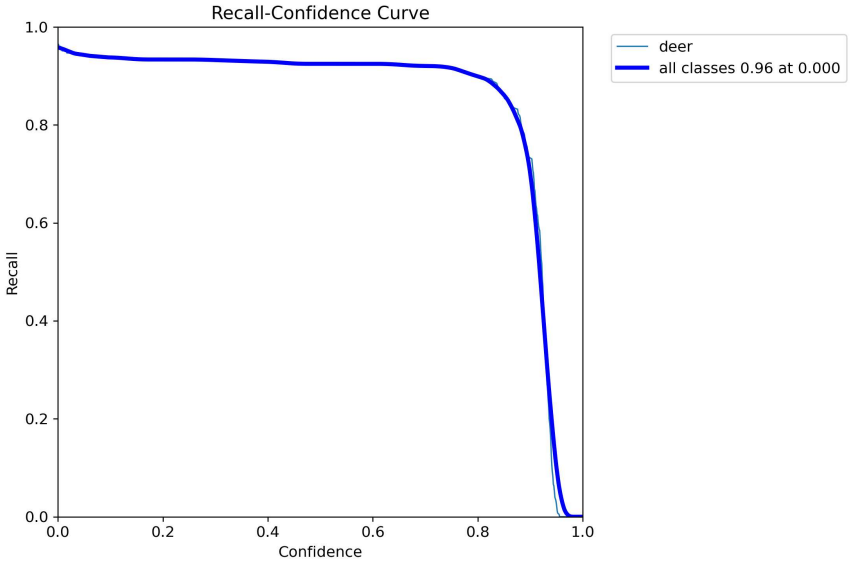

Figure 5.40: Recall curve

The final predictions made by the model are accurate and precise. The process of training turned out both efficient and successful.

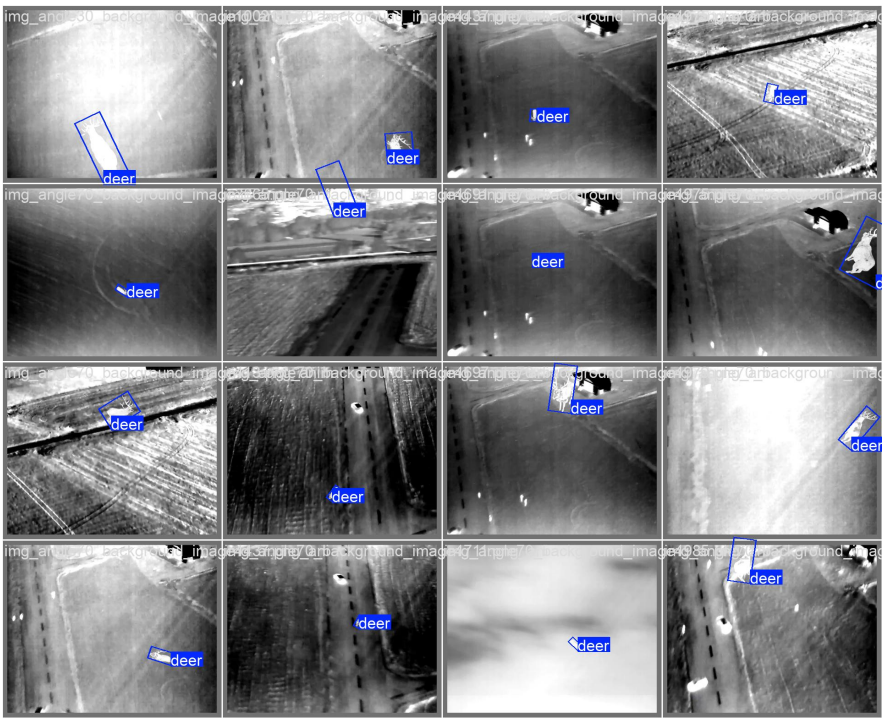

Figure 5.41: Validation batch labels 1

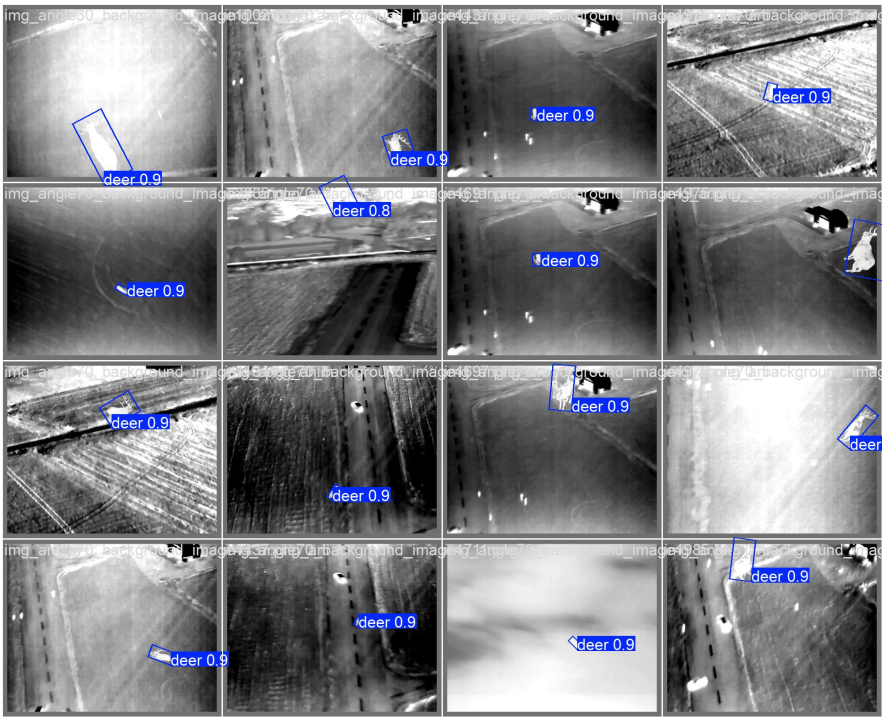

Figure 5.42: Validation batch predictions 1

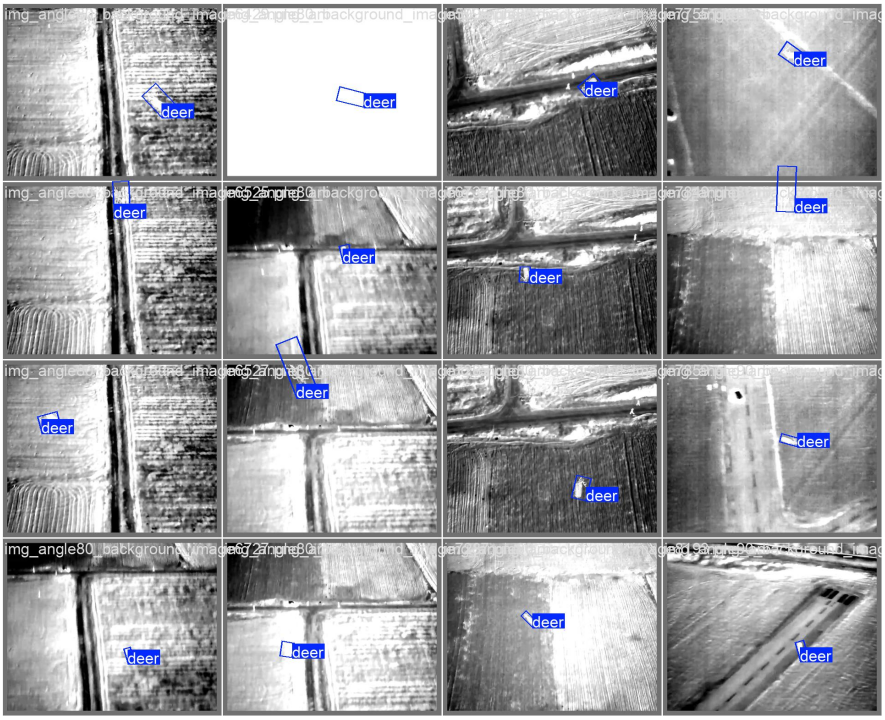

Figure 5.43: Validation batch labels 2

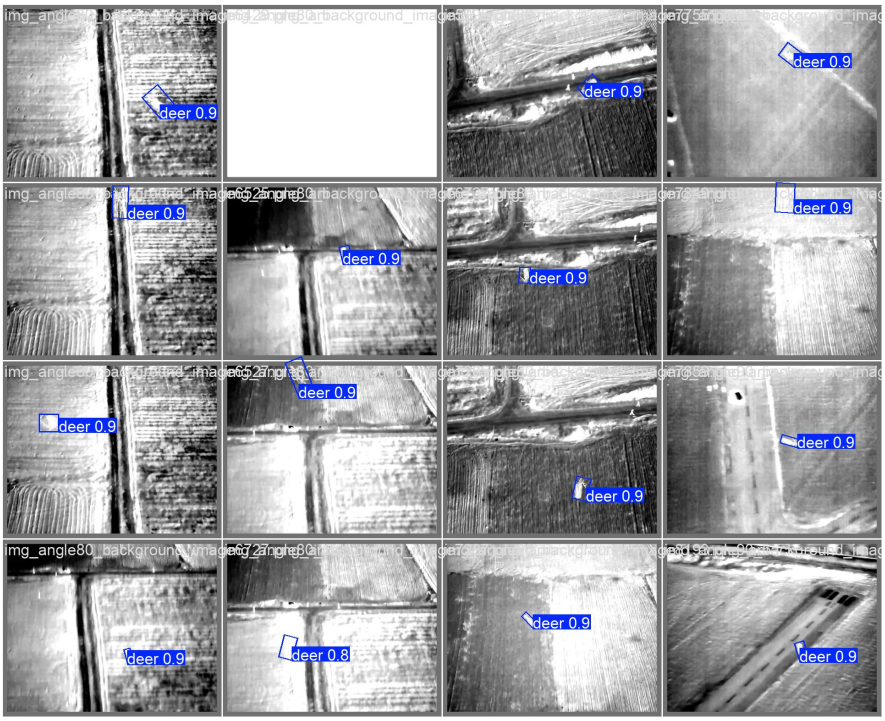

Figure 5.44: Validation batch predictions 2

Here are the results of testing the model on unseen data which is in the test folder.

Figure 5.45: Image at 40 degrees with detected object

Figure 5.46: Image at 50 degrees with detected object

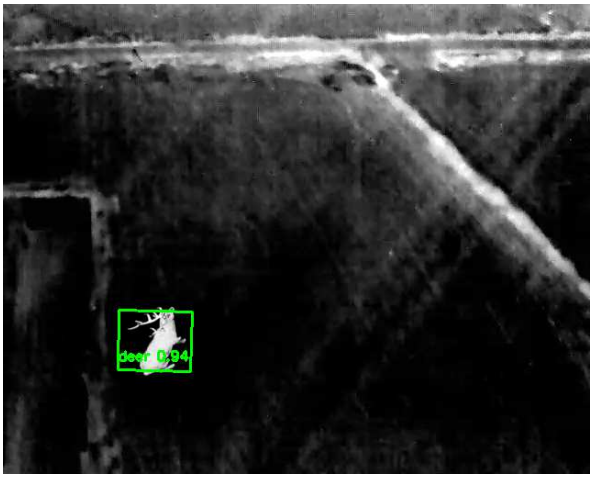

Figure 5.47: Image at 60 degrees with detected object

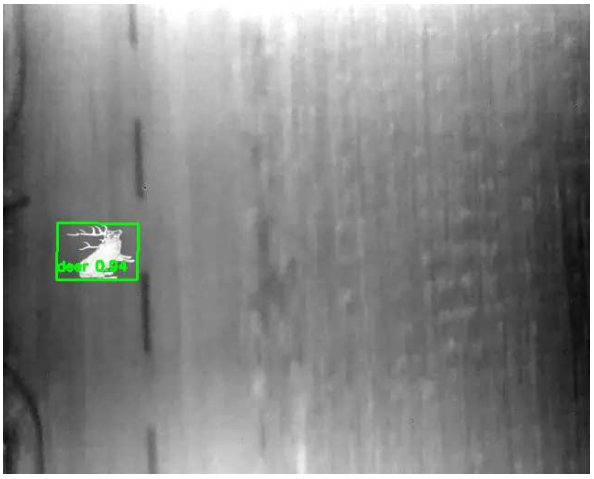

Figure 5.48: Image at 70 degrees with detected object

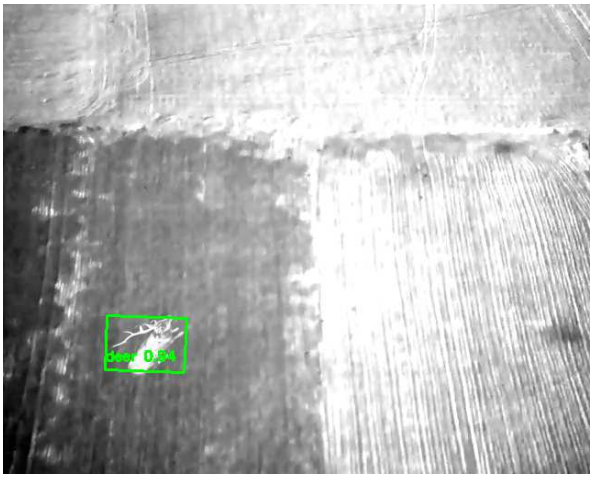

Figure 5.49: Image at 80 degrees with detected object

6 Conclusion

Synthetic data is an advantage in itself. The speed and simplicity of their collection, which actually only involves precise manufacturing, is their first advantage. The second advantage is definitely the results that can be achieved using them. When thermal image is used as synthetic data, a set can be obtained that is extremely favorable for use and processing. In the paper, two sets of synthetic data were created as an upgrade of the already existing ones. Synthetic thermal object was added to the thermal images of the datasets. The detection used on them is implemented in the form of a python script in the Scripting element in Blender. The two types of detection are implemented - axis-aligned and orienting. Using object-independent logic, the detection code is applicable to use on any object placed in the scene. The algorithm for detection is focused on the meshes and vertices of the imported object, which makes the algorithm useful to detection a wide range of possible objects. The YOLOv8 model was used for training, where the synthetic data gave enviable results for the datasets. The synthetic thermal element added to the image shows better results than the already existing real objects, proving the accuracy and success of the synthetic data. The detection results on the previously unseen test data set depending on how the model was trained also show exceptional high accuracy results.

7 Future Work

Considering the great applicability of this solution, future work can have its impact in many branches and industries. For detection on thermal synthetic images, the most potential is seen in the detection of objects and movements on the sea and waters, especially in night conditions and in general low light conditions. Also, the direction in which this could be further developed is the safe and secure rescue of people in unsuitable places such as: mountains, caves, abysses and the like. The use of the developed detection algorithm would facilitate the detection of people and speed up the search process in favor of rescue. Also, this solution is applicable in smart agriculture as well, which was demonstrated by supplementing the MONET dataset with a deer as an example of an animal that potentially needs to be monitored and alerted, whether it is for or against it.

References

[1] Ultralytics, “Oriented bounding boxes documentation,” https://docs.ultralytics.com/datasets/obb/, 2023, accessed: 2023-07-17.

[2] J. Suo, T. Wang, X. Zhang, H. Chen, W. Zhou, and W. Shi, “Hit-uav: A high-altitude infrared thermal dataset for unmanned aerial vehicle-based object detection,” *Scientific Data*, vol. 10, no. 227, pp. 1–9, July 2023. https://doi.org/10.1038/s41597-023-02066-6

[3] L. Riz, A. Caraffa, M. Bortolon, M. L. Mekhalfi, D. Boscaini, A. Moura, J. Antunes, A. Dias, H. Silva, A. Leonidou, C. Constantinides, C. Keleshis, D. Abate, and F. Poiesi, “The monet dataset: Multimodal drone thermal dataset recorded in rural scenarios,” 2023. [Online]. Available: https://arxiv.org/abs/2304.05417

[4] D. Broyles, C. R. Hayner, and K. Leung, “Wisard: A labeled visual and thermal image dataset for wilderness search and rescue,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 9467–9474. https://doi.org/10.1109/IROS47612.2022.9981298

[5] A. Vidhya, “Mosaic data augmentation,” https://www.analyticsvidhya.com/blog/2023/12/mosaic-data-augmentation/, December 2023, accessed: 2023-07-14.

[2] [3] [4] [5]

Abstract

Oriented Object Detection in Thermal Aerial Images from Drones Based On Synthetic Data and Neural Networks

Andreja Jurasović

The aim of this work is to prove the advantages of synthetic thermal images as a data set on which a detection model can be trained. A synthetic thermal object - a drone and a deer - was introduced into the already existing thermal images, and an algorithm for their axis-aligned and oriented detection was created. This resulted in annotations of new added elements on which, in addition to already existing annotations, the YOLOv8 model was trained. The success of the detection is high and is also shown by the detection made on the test data set, which was previously unseen by the model.

Keywords: UAV; thermal imaginig; aerial imaging; neural networks; synthetic data; datasets; object detection; axis-aligned bounding box; oriented boundning box; segmentation

Sažetak

Orijentirana detekcija objekata na termalnim slikama iz bespilotnih letjelica na temelju sintetičkih podataka i neuronskih mreža

Andreja Jurasović

Cilj ovog rada je dokazati prednosti sintetičkih termalnih slika kao skupa podataka na kojima se može trenirati model detekcije. U već postojeće termalne slike uveden je sintetički termalni objekt - UAV i jelen te je napravljen algoritam za njihovu osno usmjerenu i orijentiranu detekciju. To je rezultiralo anotacijama novih dodanih elemenata na kojima je, uz već postojeće anotacije, treniran YOLOv8 model. Uspješnost detekcije je visoka što pokazuje i detekcija napravljena na testnom setu podataka koji prethodno nije bio vidljiv na modelu.

Ključne riječi: UAV; termalne slike; slike s bespilotne letjelice; neuronske mreže; sintetički podaci; baza podataka; odetekcija objekata; granični okvir poravant po osi; orijentirani granilni okvir; segmentacija