Survey about object detection using deep neural networks

Dimitri Ngatcha Pokouane, dn92@tu-clausthal.de
TU Clausthal, Germany

***Abstract*—Object detection is a computer vision technique that identifies, classifies and locate a particular object in a particular setting. It has become increasingly important in recent years due to its wide range of applications, including advanced driver assistance systems (ADAS), video surveillance, and image retrieval systems. In this report, we will provide an overview of common state-of-the-art object detectors and highlight differences in approach. We will also discuss the practical part of our project, including how we trained our object detector to recognize airplanes using a dataset we created. Our project aims to develop an object detection system that can accurately detect airplanes in satellite images. We will use a deep learning approach to train our model and evaluate its performance using various metrics such as precision, recall, and F1 score.**

I. Introduction

Object detection is a crucial task in computer vision that involves identifying and localizing objects within an image or video. It is a fundamental problem in many real-world applications, such as surveillance, autonomous driving, and robotics. In recent years, object detection has seen significant advances due to the advent of deep learning, especially Convolutional Neural Networks (CNNs). CNN-based object detectors have achieved state-of-the-art performance on various datasets and tasks [1]. The aim of this project is to explore the field of object detection, with a focus on state-of-the-art object detectors. We will begin by providing an overview of four popular object detectors and highlighting the differences in their approach. Then, we will move on to the practical part of the project, where we will train an object detector of our choice to recognize airplanes. To accomplish this, we will first create a dataset using the OpenImage dataset[2], and then train a network using the selected object detector. Finally, we will present a demonstrator application to showcase the capabilities of the trained object detector. This report is organized as follows. In section II, we are going to provide the basics of object detection. In Section III, we will provide an overview of the state-of-the-art object detectors. In Section IV, we will describe our methodology for training an object detector to recognize airplanes. In Section V, we will present the results of our experiments, including an evaluation of the trained object detector's performance. Finally, we will conclude the report in Section VI , summarizing our findings and discussing potential avenues for future work.

II. The basics of object detection

–**Object detection** is the task of identifying and localizing objects of interest within an image or video. It is a critical component of many computer vision applications, including self-driving cars, robotics, and surveillance [3]. Object detection has two main components: object classification and object localization.

–**Object classification** is the process of determining the class or category of an object within an image. This can be done using a variety of machine learning techniques, including deep learning neural networks. In object detection, object classification is typically done using a pre-trained neural network that has been trained on a large dataset of images.

–**Object localization** is the process of identifying the location of an object within an image. This is typically done using bounding boxes, which are rectangular regions that tightly enclose the object. Bounding boxes can be generated using a variety of techniques, including sliding window approaches, region-based approaches, and anchor-based approaches.

One popular approach to object detection is **region-based object detection**. In this approach, a set of candidate object regions is generated using a proposal mechanism, such as selective search or region proposal networks [4]. These candidate regions are then classified and refined using a convolutional neural network (CNN). CNNs are a popular deep learning technique for object detection because they can automatically learn high-level features from raw image data. CNNs typically consist of several convolutional layers, followed by a series of fully connected layers. During training, the CNN learns to identify and classify objects by adjusting the weights of the connections between the layers.

–**State-of-the-art object detection systems**, such as Faster R-CNN [5] and YOLO [1], use CNNs to achieve high accuracy and efficiency. These systems incorporate a variety of advanced techniques, such as feature pyramid networks, anchor-based proposals, and non-maximum suppression, to further improve performance. Feature pyramid networks are a way of combining low-level and high-level features from different layers of the CNN to enhance the detection of objects at different scales. Anchor-based proposals are predefined bounding boxes that are used as reference points for predicting the location and size of the objects. Non-maximum suppression is a post-processing step that

eliminates redundant or overlapping bounding boxes and keeps only the ones with the highest confidence scores.

III. COMMON TYPES OF OBJECT DETECTORS

In this section, we will explore different types of object detectors, including region-based, one-shot, and anchor-free detectors, and explain their approach and performance differences.

A. *Region-based Detectors*

Region-based detectors are one of the most popular object detection methods. They work by first generating a set of region proposals, which are potential object bounding boxes, and then classifying each proposal as either an object or background. Region-based detectors can be divided into two main categories: two-stage and single-stage detectors. Two-stage detectors, such as Faster R-CNN [6], R-FCN [7], and Mask R-CNN [8], consist of a region proposal network (RPN) followed by a classification network. The RPN generates region proposals and then the classification network refines and classifies each proposal.

Single-stage detectors, such as YOLO [1], SSD [9], and RetinaNet [10], perform object detection in a single pass through the network. They divide the input image into a set of grids and predict the object's class and bounding box directly from each grid.

B. *One-shot Detectors*

One-shot detectors, such as Single Shot Detector (SSD)[9] and YOLOv2 [1], perform object detection using a single convolutional neural network (CNN). They work by dividing the input image into a grid and predicting object classes and bounding boxes for each grid cell.

C. *Anchor-free Detectors*

Anchor-free detectors, such as CenterNet [11], FCOS [12], and CornerNet [13], are a relatively new approach to object detection. They eliminate the need for predefined anchors used in the region-based detectors and predict the center and size of each object directly. Anchor-free detectors have shown promising results, especially in detecting small objects and objects with irregular shapes.

D. *Cascade Detectors*

Cascade detectors, such as Cascade R-CNN(Region-based Convolutional Neural Networks) , use a cascade of neural networks to refine object proposals. Each network in the cascade improves the proposals generated by the previous network, leading to higher detection accuracy[14].

E. *Unified Two-Stage Detectors*

Unified two-stage detectors represent a novel category of object detection architectures that combine the strengths of traditional two-stage detectors and one-shot detectors within a single framework. Unlike the conventional approach of employing separate networks for region proposal and object classification, unified two-stage detectors streamline this process by utilizing a single network for both tasks. This integration offers notable computational efficiency and memory utilization benefits. A common practice in two-stage detection, exemplified by models like Faster R-CNN, involves distinct networks for proposal generation and subsequent object classification. In contrast, unified two-stage detectors employ a shared backbone and feature extractor to handle both region proposal and classification tasks.

Several notable instances of unified two-stage detectors include:

-Cascade R-CNN : This architecture adopts a cascaded approach by utilizing multiple R-CNN heads with progressively increasing Intersection over Union (IoU) thresholds. This iterative refinement enhances region proposals and augments detection accuracy.

-Libra R-CNN [15]: Addressing the challenge of class imbalance in object detection, Libra R-CNN introduces a balanced feature pyramid, a sampling strategy that maintains balance, and a corresponding loss function. These components collectively mitigate the class imbalance issue.

-TridentNet [16]: TridentNet introduces a trident convolution module to generate multi-scale feature maps, which facilitates object detection across various scales. This approach enhances the network's capability to identify objects of differing sizes.

This emerging paradigm of unified two-stage detectors represents a fusion of the conventional two-stage and one-shot architectures, offering advantages in terms of efficiency, memory usage, and detection accuracy.

In summary, object detection has advanced significantly in recent years with the development of various approaches, including region-based, one-shot, anchor-free, cascade, and two-stage one-shot detectors. Overall, the choice of object detector depends on the specific application and trade-offs between accuracy and speed. It is important to consider the strengths and weaknesses of each type of detector before selecting one for a particular task.

IV. STATE-OF-THE-ART OBJECT DETECTORS

"In recent years, deep learning methods have achieved state-of-the-art results in various computer vision tasks, including object detection."[17] Deep learning methods have shown remarkable performance in object detection, and they have become the standard approach for many applications.

Convolutional Neural Networks (CNNs) [18] are the most commonly used deep learning method for object detection. CNNs are a type of feedforward neural network that use convolutional layers to automatically learn hierarchical representations of features in images. These learned features can then be used for object detection.

There are two main types of CNN-based object detectors: region-based and one-shot. Region-based detectors, such as Faster R-CNN and Mask R-CNN, first generate a set of candidate object regions and then classify them as object or background. One-shot detectors, such as YOLO and SSD, use

a single network to simultaneously predict object locations and classifications.

–**Faster R-CNN** is currently the state-of-the-art region-based object detector. It uses a region proposal network to generate object region proposals and a separate classification network to classify these regions. Mask R-CNN extends Faster R-CNN by adding a segmentation network that produces object masks in addition to object bounding boxes and classifications.

–**YOLO** (You Only Look Once) : Is one popular one-shot object detectors. YOLO uses a single CNN to predict object classes and bounding boxes directly from the input image [1]. YOLO divides the input image into a grid and predicts object probabilities and bounding boxes for each cell in the grid. YOLO is known for its speed, but it may not perform as well as other methods in detecting small objects. YOLOv1 was introduced by Joseph Redmon et al. in 2015 as a fast and accurate object detection model that predicts bounding boxes and class probabilities in a single pass. It divided the input image into a grid and used a convolutional neural network (CNN) to output a vector for each grid cell. The vector contained the coordinates, dimensions, confidence and class probabilities of the bounding box for that cell.
***YOLOv1** achieved 45 frames per second (fps) on a GPU and had less false positives on the background than other models at that time[19].
***YOLOv2** was released in 2017 as an improvement over YOLOv1. It introduced several techniques to enhance the speed and accuracy of the model, such as batch normalization, anchor boxes, dimension clusters, fine-grained features, multi-scale training and testing, and a new network architecture called Darknet-19. YOLOv2 achieved 78.6 mAP (mean average precision) on the Pascal VOC 2007 dataset at 40 fps[19].
***YOLOv3** was launched in 2018 as an incremental improvement over YOLOv2. It used a deeper network architecture called Darknet-53, which had more layers and residual connections. It also added an objectness score to the bounding box prediction and used feature maps from different layers of the network to make predictions at three different scales. This improved the performance on small objects. YOLOv3 achieved 57.9 mAP on the COCO dataset at 20 fps[19].
***YOLOv4** was released in 2020 by a different team of authors than the previous versions. It incorporated many innovations from recent research papers to optimize the speed and accuracy of object detection. Some of these innovations included Mosaic data augmentation, DropBlock regularization, Cross Stage Partial Network (CSPNet), Self-Adversarial Training (SAT), Mish activation function, Spatial Pyramid Pooling (SPP). YOLOv4 achieved 43.5 mAP on the COCO dataset at 65 fps[19].
***YOLOv5** was developed by Ultralytics in 2020 as a continuation of YOLOv4. He improved the model's performance and added new features such as hyperparameter optimization, integrated experiment tracking and automatic export to popular export formats. ***YOLOv6** was open-sourced by Meituan in 2022 and is in use in many of the company's autonomous delivery robots.
***YOLOv7** added additional tasks such as pose estimation on the COCO keypoints dataset.
***YOLOv8** is the latest version of YOLO by Ultralytics, released on January 10, 2023. It builds on the success of previous versions and introduces new features and improvements for enhanced performance, flexibility, and efficiency. YOLOv8 supports a full range of vision AI tasks, including detection, segmentation, pose estimation, tracking, and classification. It uses a new backbone network called FastSAM (Fast Self-Attention Module), which combines convolutional layers and self-attention mechanisms in a novel way. It also uses a new loss function called Focal Loss with Dynamic Alpha (FLDA), which adapts the loss weight according to the difficulty of each sample. YOLOv8 achieves 54.2 mAP on the COCO dataset at 160 fps [20].

–**Single Shot Detectors**(SSD): Is a real-time object detection algorithm that uses a single CNN to predict object classes and bounding boxes directly from the input image [9]. It uses multiple feature maps with different resolutions to detect objects at different scales. The network predicts object classes and locations simultaneously using a set of default bounding boxes of various aspect ratios and scales. SSD is a popular method because of its speed and accuracy.

–**RetinaNet:** Is a state-of-the-art object detection method that addresses the problem of class imbalance in object detection [10]. It uses a novel focal loss function that gives more weight to hard examples during training, resulting in better performance on small objects. RetinaNet uses a feature pyramid network (FPN) to detect objects at different scales and achieves state-of-the-art results on several benchmarks. Other deep learning methods for object detection include fully **convolutional networks (FCNs)**, which can perform object detection in a single pass, and attention-based methods, which use attention mechanisms to focus on relevant regions of the image. A recent example of this approach is DETR [21], which uses a transformer-based architecture to encode the global context of the image and decode the object locations and classes. Transformer-based architectures are currently achieving major strides in AI technology and have shown promising results in object detection. However, at the moment they are not as widely used as traditional approaches, such as RetinaNet or Faster R-CNN.

Object detection is a fundamental task in computer vision that aims to locate and classify objects within an image. Over the years, several approaches have been proposed to solve this task, ranging from traditional methods to more recent deep learning-based methods. We will compare here the properties and performance some of the state-of-the-art object detectors, such as **YOLO, SSD, Faster R-CNN, and RetinaNet**.

COMPARISON OF OBJECT DETECTORS

Detector	Advantages	Disadvantages	Comparison
YOLO	Very fast and optimized for real-time applications. Can detect many objects in a single pass. Uses only one CNN model, leading to a simple architecture.	May have difficulty detecting small objects. Less precise than other detectors.	Compared to other detectors, YOLO is the fastest and simplest, but also the least precise.
Faster R-CNN	Precise and robust detection of objects. Uses a two-stage architecture with an RPN and a network for object detection, resulting in precise localization of objects. Can also detect small objects.	Slower than other detectors. Requires training on large datasets to achieve good performance. Requires more memory and processing power than other detectors.	Faster R-CNN is a precise detector, but slower and more computationally intensive than other detectors.
SSD	Very fast and optimized for real-time applications. Uses only one CNN model, leading to a simple architecture. Can also detect small objects.	Less precise than other detectors.	SSD is a fast detector, but less precise than other detectors.
RetinaNet	Very precise detection of objects. Uses focal loss to solve the problem of uneven distribution of positive and negative examples in object detection. Can detect objects of different sizes well.	Slower than other detectors. Requires training on large datasets to achieve good performance. Requires more memory and processing power than other detectors.	RetinaNet is a precise detector, but slower and more computationally intensive than other detectors.
Mask R-CNN	Can not only detect but also segment objects. Precise and robust detection of objects. Uses a two-stage architecture with an RPN and a network for object detection and segmentation.	The slowest detector among those mentioned here. Requires training on large datasets to achieve good performance. Requires more memory and processing power than other detectors.	Mask R-CNN is a precise detector, but slower and more computationally intensive than other detectors.

In summary, deep learning methods have revolutionized object detection and have led to significant improvements in accuracy and efficiency. Region-based and one-shot detectors are the most popular approaches, and Faster R-CNN and YOLO are among the state-of-the-art methods. Each method has its strengths and weaknesses and may be better suited for different applications depending on the specific requirements. The choice of the method also depends on factors such as speed, accuracy, and the available hardware.The next section will focus on the practical implementation of object detection, specifically training an object detector to recognize airplanes using a custom dataset.

V. Training an Object Detector for Aircraft Recognition

In this section, we will discuss the practical part of the project, which involves training an object detector to recognize airplanes. We will explain how to create a dataset for training the object detector and how to train a CNN for airplane detection. Finally, we will present the demonstrator application.

A. Creating a Dataset

Before we will train the object detector, we need to create a dataset of images containing airplanes. In our case, we choose to use OpenImage dataset [22], which contains over nine million images across thousands of classes, including airplanes. We can download the images and use them to create a new dataset for airplane detection.

B. Training a CNN

Once we have a suitable dataset, we will start training a CNN for airplane detection. In our case, we will be using the YOLOv8 framework, which is the latest and most advanced version of the YOLO (You Only Look Once) model for fast and accurate object detection. YOLOv8 uses a single neural network to predict bounding boxes and class probabilities directly from full images, making it ideal for real-time applications. YOLOv8 also supports a full range of vision AI tasks, such as segmentation, pose estimation, tracking, and classification, offering unparalleled performance in terms of speed and accuracy [20]. To train the YOLOv8 network, we need to first initialize the weights using a pre-trained model. We can then fine-tune the network on our airplane dataset. The fine-tuning process involves adjusting the weights of the network to minimize the loss function, which measures the difference between the predicted and ground-truth bounding boxes.

Once the training is complete, we will evaluate the performance of our network on a test dataset. We will use metrics such as precision, recall, and F1 score to measure the accuracy of our network.

C. Demonstrator Application

The training process involved using the Airplane dataset from Openimage, with 1305 images for fast training and YOLOv8 for 300 epochs. After training, 400 images were used for validation to evaluate the model's performance. Google Colab and GPU were utilized to expedite the training process. The final results indicate that the model achieved promising performance, with high precision, recall, and mAP

values during validation. The early stopping mechanism helped stop training when no improvement was observed in the last 50 epochs, and the best model was saved at epoch 198. After training, we get the result below of the parameter.

Overall Result

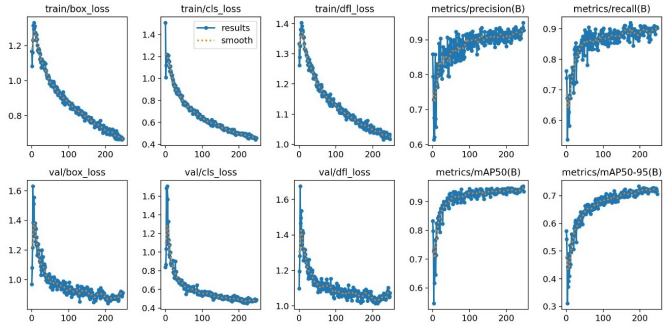

Fig. 0: Result of training

***Training Metrics:**

-train/boxLoss: The curve shows a gradual decrease from to rapidly approaching 0, indicating a significant reduction in the bounding box loss during training.

-train/clsLoss: The curve exhibits a consistent decline from 1.55 to almost 0, indicating a substantial reduction in the classification loss during training.

-train/dflLoss: The curve demonstrates a gradual decline from 1.4 to approaching 0, indicating a significant decrease in the deformable convolutional layer (DFL) loss during training.

-metrics/precision(B): The curve displays a steady increase from 0.65 to close to 1, indicating a substantial improvement in precision for the positive class (B) during training.

-metrics/recall(B): The curve shows an increase from 0.2 to approaching 0.94, indicating a significant enhancement in recall for the positive class (B) during training.

***Validation Metrics:**

-val/boxLoss: The curve depicts a decrease from 1.65 to around 0.3, indicating a notable reduction in the bounding box loss during validation.

-val/clsLoss: The curve displays a decline from 1.65 to approximately 0.45, indicating a significant reduction in the classification loss during validation.

-val/dflLoss: The curve illustrates a decline from 1.65 to around 1.08, indicating a decrease in the deformable convolutional layer (DFL) loss during validation.

-metrics/mAP50(B): The curve demonstrates a plateau from 0 to close to 0.95, indicating a high mean Average Precision (mAP) at 50% overlap for the positive class (B) during validation.

-metrics/mAP50-95(B) The curve shows a plateau from 0.33 to around 0.75, indicating a high mean Average Precision (mAP) from 50% to 95% overlap for the positive class (B) during validation.

***Analysis:**

The training curves for box_loss, cls_loss, and dfl_loss demonstrate a gradual decline, indicating that the model is effectively learning and reducing the respective losses during training. This reduction suggests that the model is improving and refining its ability to localize and classify objects accurately.

The metrics precision(B) and recall(B) show positive trends, with precision(B) nearing perfection and recall(B) approaching a high value, indicating that the model can both accurately classify and capture most positive instances during training.

During validation, the curves for box_loss, cls_loss, and dfl_loss also demonstrate reductions, suggesting that the model generalizes well to unseen data and maintains its ability to localize and classify objects accurately.

The high mAP values for both mAP50(B) and mAP50-95(B) during validation indicate that the model performs remarkably well across different overlap thresholds, achieving high precision and recall levels on unseen data.

***Conclusion:**

The observed performance metrics and their trends indicate that your YOLOv8 model has been successful in learning meaningful features and generalizing well to unseen data. The high precision, recall, and mAP values attest to its efficacy in accurately detecting and classifying objects, making it a strong candidate for real-world applications.

Let's now analyse the confusion matrix.

Confusion matrix

The confusion matrix is a vital tool that allows us to evaluate and understand how our trained model performs on different classes within the training set. It presents a tabular representation of predicted labels versus actual labels for each class, enabling us to assess the model's accuracy and identify potential areas of improvement. By analyzing the confusion matrix, we gain insights into how well the YOLOv8 model has learned to differentiate between various classes in the Airplane dataset. Each row in the matrix corresponds to the true class, while each column corresponds to the predicted class. The diagonal elements represent the number of correct predictions, indicating instances where the model successfully recognized the class. Off-diagonal elements represent misclassifications, where the model made errors in predicting the class. Having this information empowers us to fine-tune the model and address any challenges it faces when dealing with specific classes. It guides us in making informed decisions to enhance the model's overall performance and accuracy on the training set.

The picture below shows the confusion matrix of training set

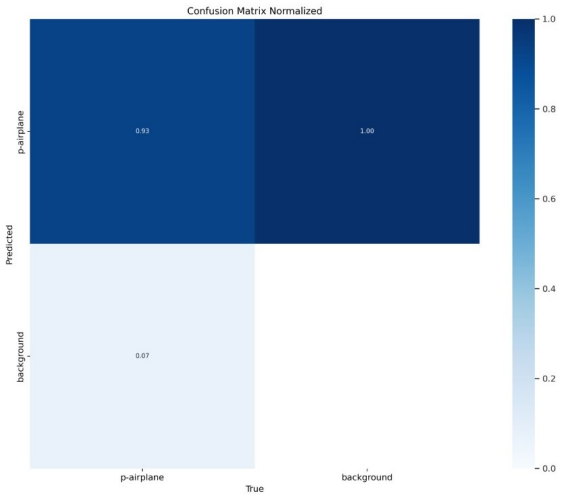

Fig. 1: Confusion matrix of training set

-Class Labels:

”p-airplane”: This refers to the class label ”airplane” that the model is attempting to predict. ”background”: This refers to the class label ”background,” representing areas in the images that do not contain airplanes. True Positives (TP) and False Positives (FP):

”p-airplane” (Predicted Airplane): The model correctly predicted 93% of the instances labeled as ”airplane” in the training set. These are the true positives (TP) for the ”p-airplane” class. ”background”: The model made false positive predictions for 7% of the instances labeled as ”background” in the training set. These are the false positives (FP) for the ”p-airplane” class. True Negatives (TN) and False Negatives (FN):

”background”: The model correctly predicted 100% of the instances labeled as ”background” in the training set. These are the true negatives (TN) for the ”background” class. ”p-airplane” (Predicted Airplane): There is no information provided for the true negatives and false negatives for the ”background” class. Accuracy and Misclassification:

The accuracy of the model on the training set can be calculated as the sum of true positives and true negatives divided by the total number of samples. However, the information given in the confusion matrix is not sufficient to compute the overall accuracy.

-Interpretation:

The model exhibits a high true positive rate (93%) for the ”airplane” class, which indicates that it is proficient at correctly identifying airplanes in the images. The false positive rate (7%) for the ”airplane” class implies that there are instances where the model incorrectly predicts an object as an airplane when it’s actually part of the background. This could be an area for improvement. ****F1-Score for Training and Validation Sets***

The F1-score is a crucial performance metric used to evaluate the effectiveness of our trained YOLOv8 model on both the training and validation datasets. It is especially useful when dealing with imbalanced datasets, where one class may dominate the samples more than others. A high F1-score on the training set signifies that the YOLOv8 model successfully identifies instances of airplanes (positive class) while effectively minimizing false positives (incorrectly predicting airplanes in background regions). This demonstrates the model’s proficiency in generalizing to the training data, striking a good balance between accurate airplane detection and avoiding misclassifications. Moreover, a high F1-score on the validation set indicates that the model’s performance extends beyond the training data, showcasing its ability to generalize well to new and unseen instances of both ”airplane” and ”background” classes. This highlights the model’s capacity to learn meaningful features from the training set and its potential applicability in real-world scenarios.

F1-Score for the Training Set: The F1-score on the training set measures the model’s ability to balance precision and recall. Precision represents the proportion of true positive predictions out of all positive predictions, while recall (also known as sensitivity or true positive rate) represents the proportion of true positive predictions out of all actual positive samples. The F1-score is the harmonic mean of precision and recall, providing a single value that combines both measures.

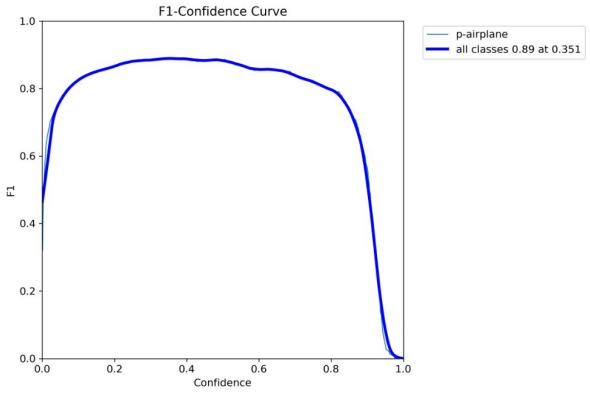

Fig. 2: F1-Score for Training set

F1-Score for the Validation Set:

The F1-score on the validation set is even more crucial as it assesses the model’s ability to generalize to unseen data. This metric determines how well the model performs on new, previously unseen images. To compute the F1-score for the validation set, the model’s predictions on the validation data are compared to the ground truth labels.

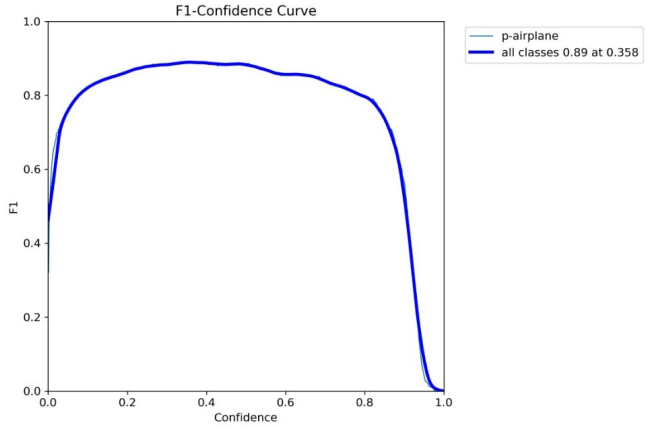

Fig. 3: F1-Score for Validation set

The F1-score for the "p-airplane" class indicates that our model performs well in identifying airplanes, achieving an F1-score of 0.89 on both the training and validation sets.

****Recall for Training and Validation Sets***

Recall is a critical performance metric used to assess the effectiveness of your YOLOv8 model on both the training and validation datasets. It measures the proportion of true positive predictions out of all actual positive samples, providing valuable insights into how well the model captures positive instances, such as detecting airplanes correctly.

Recall for the Training Set: The recall on the training set demonstrates how well your model is able to identify instances of airplanes within the training data. A high recall indicates that the model successfully captures a significant number of actual airplanes present in the images during training. It reflects the model's ability to learn and recognize patterns specific to the "airplane" class, contributing to its overall performance.

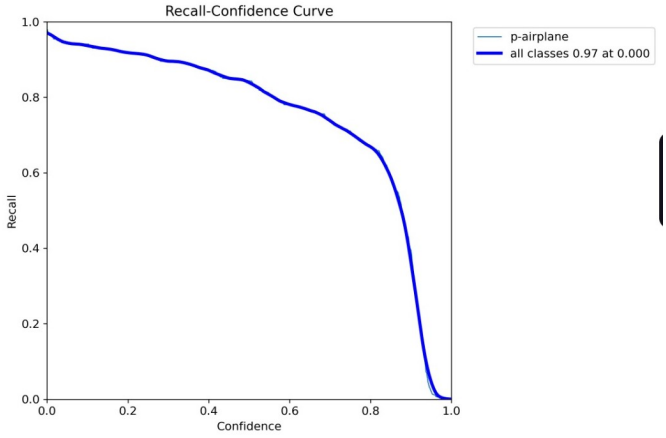

Fig. 4: Recall for the Training Set

Recall for the Validation Set: The recall on the validation set is of paramount importance as it evaluates the model's generalization capability to new, unseen data. A high recall on the validation set indicates that the model can effectively detect airplanes in previously unseen images. It reflects the model's capacity to learn meaningful features during training and apply that knowledge to correctly identify "airplane" instances in real-world scenarios.

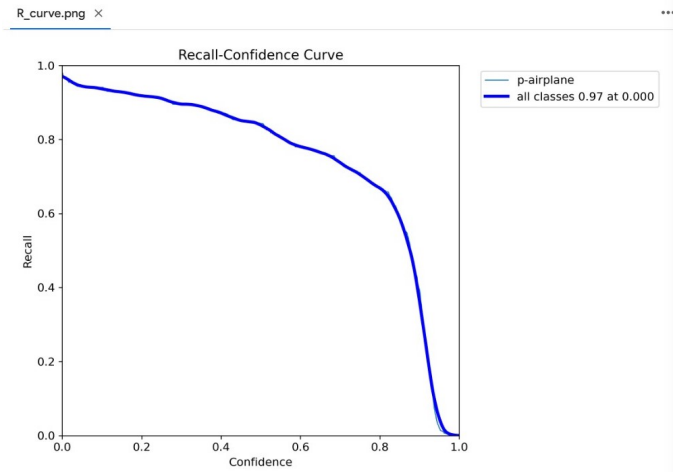

Fig. 5: Recall for the Validation Set

These recall scores indicate that your YOLOv8 model is performing exceptionally well in capturing positive instances of airplanes in both the training and validation datasets. With a recall of 0.97 for all classes, including the background class, the model demonstrates a high ability to correctly identify airplanes while minimizing false negatives (instances where airplanes were missed). A recall of 0.97 means that your model is correctly identifying 97% of the actual airplanes in the dataset. This high recall score suggests that our model is proficient at recognizing the patterns and features associated with airplanes.

Precision for Training and Validation Sets

Precision is a crucial performance metric used to evaluate the effectiveness of your YOLOv8 model on both the training and validation datasets. It measures the proportion of true positive predictions out of all positive predictions made by the model, providing valuable insights into the accuracy of positive predictions, such as correctly identifying airplanes.

Precision for the Training Set: The precision on the training set indicates how well your model is performing in terms of making accurate positive predictions within the training data. A high precision score suggests that when the model predicts an instance as an airplane, it is more likely to be correct. It reflects the model's ability to learn and distinguish patterns specific to the "airplane" class, contributing to its overall performance.

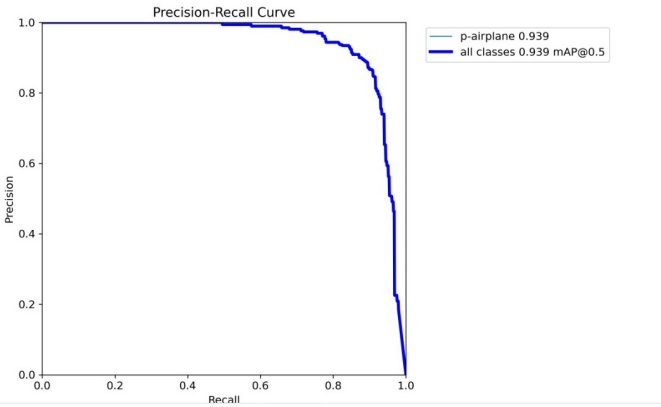

Fig. 6: Precision for the Training Set

Precision for the Validation Set: The precision on the validation set is equally important as it evaluates the model's ability to generalize to new, unseen data. A high precision score on the validation set indicates that when the model makes predictions on previously unseen images, it maintains a high level of accuracy in identifying airplanes. This showcases the model's capacity to learn meaningful features during training and apply that knowledge accurately in real-world scenarios.

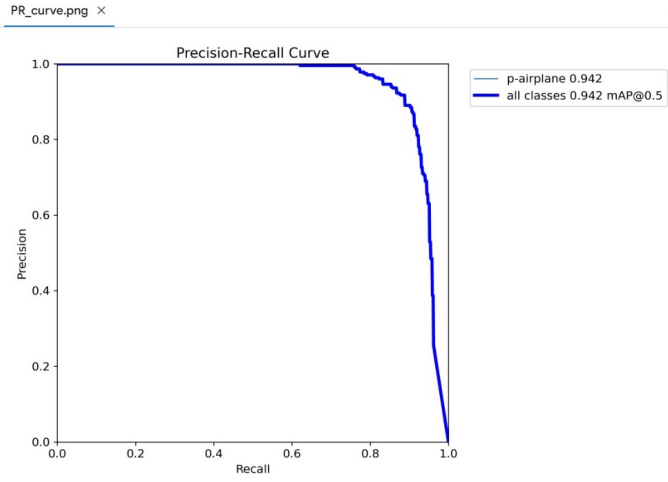

Fig. 7: Precision for the Validation Set

These precision scores indicate that our YOLOv8 model is performing exceptionally well in making accurate positive predictions for both the "airplane" class and all classes combined (using the mean precision at a 0.5 threshold). A precision of 0.939 (or 93.9%) means that when your model predicts an instance as an airplane, it is correct around 93.9% of the time. This demonstrates the model's accuracy in identifying airplanes and minimizing false positives.

***performance of our detector in detecting an airplane on a picture**

Fig. 8: performance of our detector

Our detector exhibits remarkable proficiency in identifying airplanes within individual images, as evidenced by achieving an impressive confidence score of 0.91 for the 'p-airplane' class. This confidence score of 0.91 implies the model's high degree of confidence in its prediction, indicating an estimated 91% likelihood of an airplane being present in the specific image under consideration. It is worth noting, however, that while this result is valuable for illustrative purposes, it does not guarantee the model's ability to generalize equally effectively across diverse images. While the detector's accuracy and precision in airplane recognition are noteworthy within the context of this specific image, it's essential to recognize that the performance might differ when applied to a wider range of images and scenarios. Therefore, while our detector exhibits promise and potential, its true effectiveness should be evaluated through comprehensive testing on diverse datasets that encompass various environmental conditions, lighting scenarios, and object orientations.

VI. Conclusions and Outlook

This research created a detailed review of common state-of-the-art object detectors. We were able to find differences in accuracy, speed, ability to process many objects, complexity of the training procedure, and network size by selecting four detectors with different approaches. Faster R-CNN, SSD, YOLO, RetinaNet, and Mask R-CNN were the detectors chosen. This summary offered useful information on the current state-of-the-art in object detection.

Object detection is a rapidly evolving research area that continues to make significant advances. In the future, detector accuracy could be improved through the use of more advanced architectures and training methods. Additionally, detectors could be able to perform multiple tasks such as object tracking and segmentation. Another important topic is the improvement of data augmentation and pre-processing to increase the quality and quantity of training data. Finally, object detection could be further applied in various applications such as the automotive industry, surveillance systems, and robotics. Overall, object detection offers a wide range of applications and a promising future. This master project has contributed to deepening the

understanding of this technology and gaining valuable insights into common detectors.

REFERENCES

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[2] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, and V. Ferrari, "The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale," *arXiv preprint arXiv:1811.00982*, 2018.

[3] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[4] J. R. R. Uijlings, K. E. A. Van De Sande, T. Gevers, and A. W. M. Smeulders, "Selective Search for Object Recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013.

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 91–99.

[6] S. Ren, K. He, R. Girshick, and Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[7] J. Dai, Y. Li, S. Kichenko, F. Zhou, and J. Lu, "R-FCN: Object Detection via Region-based Fully Convolutional Networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 379–387.

[8] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2980–2988.

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," in *European Conference on Computer Vision*, 2016, pp. 21–37.

[10] T.-Y. Lin, P. Goyal, and R. Girshick, "Focal Loss for Dense Object Detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2999–3007.

[11] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," *arXiv preprint arXiv:1904.07850*, 2019.

[12] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully convolutional one-stage object detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 9627–9636.

[13] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 734–750.

[14] Z. Cai and N. Vasconcelos, "Cascade r-cnn: High quality object detection and instance segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, p. 1–1, 2019.

[15] J. Pang, K. Chen, J. Shi, H. Feng, W. Ouyang, and D. Lin, "Libra r-cnn: Towards balanced learning for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[16] Y. Li, Y. Chen, N. Wang, and Z. Zhang, "Scale-aware trident networks for object detection," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019.

[17] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[19] Gaurav Maindola, "A brief history of yolo object detection models from yolov1 to yolov5," https://machinelearningknowledge.ai/a-brief-history-of-yolo-object-detection-models/, 2021.

[20] Ultralytics, "Home - ultralytics yolov8 docs," https://docs.ultralytics.com/, 2023.

[21] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[22] OpenImages, "Openimages dataset v6," https://storage.googleapis.com/openimages/web/index.html, 2021.