International Journal of Emerging Engineering and Technology (IJEET)
ISSN (e): 2958-3764
Volume: 3, Number: 2, Pages: 1 - 5, Year: 2024

YOLOv8 vs RetinaNet vs EfficientDet: A Comparative Analysis for Modern Object Detection

Sana Fatima[1], Najmi Ghani Haider[2], Rizwan Riaz[1]

[1] Software Engineering Department, NED University of Engineering & Technology, Karachi, 75270, Pakistan

[2] Department of Computer Science, UIT University, Karachi, Pakistan

[*]Corresponding Author: Sana Fatima (Email Address: sanafatima@cloud.neduet.edu.pk)

Received: 08-07-2024 Revised: 10-11-2024 Accepted: 20-11-2024

***Abstract*— Object detection plays a vital role in computer vision. It facilitates machines to comprehend and interpret images and videos and make decisions based on visual statistics. The search for the finest object detection algorithm continues to be an important endeavour in the area of computer vision. For this purpose, this paper includes three leading models—YOLO (You Only Look Once), RetinaNet, and EfficientDet, which are thoroughly examined and analyzed for object detection. We compare these three algorithms using the COCO dataset, which mainly comprises three categories of data, which are discussed in this paper. These techniques were examined using evaluation metrics. It helps to assess which algorithm is better for object detection. For this study, we used many AI-based libraries available in Python.**

Keywords*— Artificial Intelligence, Artificial Neural Networks, Image Processing, Object Detection

I. INTRODUCTION

Advanced object detection algorithms are taking center stage in the ever-evolving realm of computer vision. These algorithms are used to identify objects in an image or video [1], each striving to find the sweet spot between computational efficiency and accuracy. This study hones in on three notable contenders – RetinaNet, EfficientDet, and YOLO (You Only Look Once) showcasing their state-of-the-art strategies in tackling the inherent challenges of object identification tasks. Recognizing the escalating demand for dependable identification systems, it becomes necessary to thoroughly understand how these algorithms perform.

The algorithms are tested on a custom dataset that emphasizes various scenarios and object classes and has been carefully selected to mimic real-world difficulties. We seek to analyze the finer points of each algorithm's performance measures, highlighting its advantages and disadvantages through a thorough assessment. In addition to the numerical analyses, the study examines the trade-offs that are present in the model's design decisions, offering a comprehensive view of the operational dynamics. This research is very important as it specifically goes beyond object detection and provides useful information for scholars and practitioners. The results are intended to enable decision-makers to select algorithms by matching options to particular use cases. Additionally, by presenting prospective directions for advancement and innovation, the research adds to the continuing conversation on developing computer vision technologies. This study is essentially an in-depth examination of the relative subtleties of YOLO, RetinaNet, and EfficientDet because of their speed and accuracy in real-time object detection tasks.

II. RELATED WORK

N. Yadav and B. Utkarsh's research dives into the realm of object identification algorithms—namely, Single Shot Detector (SSD), Faster R- CNN, and Region-based Fully Convolutional Networks (R-FCN) using the COCO dataset[2]. The research explores how these algorithms perform in different scenarios, from everyday mobile applications to critical real-time systems like self-driving cars. Using TensorFlow for implementation and maintaining constant hardware conditions, the study focuses on key parameters such as mean Average Precision (mAP), memory usage, and testing time. Notably, SSD shines when paired with lightweight feature extractors on larger images, showcasing competitive results akin to accuracy-centric algorithms. The paper underscores the importance of tailoring model choices to specific applications. In conclusion, it suggests further exploration through model combination research to unearth optimal solutions for real-world use. All in all, this research unravels the intricate dynamics of performance for leading object detection models in the realm of computer vision.

R. Padilla et al. discussed the challenges of evaluating supervised object detection techniques, considering various metrics and datasets[3]. The research emphasizes the evolution of real-time object identification applications over time, highlighting the transformative role of deep neural networks (DNNs) in computer vision breakthroughs. The paper highlights the increasing utilization of object identification methods across industries, showcasing the revolutionary impact of DNNs in the field. Recognizing the challenges posed by variations in bounding box representation formats and metric requirements among tools, the study stresses the need for standardized benchmark datasets and evaluation metrics. The paper introduces an extended evaluation tool with 13 additional indicators and

support for various annotation formats, building upon prior research. The primary technical contributions include comprehensively describing widely used measures and detailing their mathematical foundations and practical applications. The study delves into video object detection and introduces a new spatiotemporal metric. Concluding with the release of an open-source toolbox, the paper offers a consistent and reliable approach for evaluating object detection algorithms.

M. Haris and A. Glowacz[4] discussed the automobile industries, which have developed rapidly since the first demonstration in the 1980s. S. Kuutti et al. [5] describe the rapid evolution of the automobile industry since its first demonstration in the 1980s. It delves into the crucial role that precise and timely object detection plays in the realm of automated driving and automotive safety systems. To put these algorithms to the test, the study taps into the Berkeley Deep Drive (BDD100K) dataset, evaluating five image processing heavyweights: R-FCN, Mask R- CNN, SSD, RetinaNet, and YOLOv4. The study goes a step further, focusing on various sensors crucial for environmental perception, including radar, LiDAR, and cameras. By contrasting their distinct benefits and limitations, the paper emphasizes the value of leveraging cameras to augment visual data, particularly in mitigating challenges posed by dynamic environmental conditions.

Girshick[6] studied different models from the olden days of R-CNN to Cascade R-CNN. So, R-CNN kicked things off with regions and classifications, and then SPP-Net jumped in, bringing spatial pyramid pooling to speed things up without losing accuracy. Fast R-CNN then joined, mixing up how it learns where things are and what they are through RoI pooling. Faster R-CNN changed everything with Region Proposal Networks (RPN), making it faster to propose objects. And finally, Cascade R-CNN handled IoU stuff using several regressors. Basically, these models make object detection better, fixing speed, alignment, and how they spot things. Considering the facts and comparisons given, we decided to compare YOLOv8, RetinaNet and EfficientDet. For that, this experiment was conducted.

III. MATERIALS AND METHODS

A. YOLOV8 Overview

In traditional object detection methods, images are processed sequentially, often resulting in slower performance. On the other hand, YOLO adopts a holistic image analysis approach. It divides an image into a grid, independently examining each grid cell for potential objects. This grid-based strategy allows YOLO to evaluate the entire image simultaneously, making it efficient and suitable for real-time applications. The significance of YOLO becomes particularly evident in scenarios where quick and accurate object detection is critical, such as autonomous vehicles. The algorithm's ability to analyze complex visual data allows timely decision-making, enhancing its utility in applications where immediate responses are paramount. Unlike conventional methods, YOLO's grid-based methodology facilitates a comprehensive examination of each grid cell, enabling it to capture intricate details and relationships within the image. Moreover, YOLO's versatility extends beyond just detecting objects; it can discern multiple objects within a single grid cell and categorize them accurately. This capability enhances its applicability in diverse settings, from surveillance systems to medical imaging. The algorithm's speed and accuracy make it a preferred choice for various computer vision tasks, underscoring its impact on advancing the capabilities of object detection technologies. The algorithm operates through a series of key steps:

a) Grid-based Partitioning: YOLO divides the input image into a grid, often with grid sizes like 7x7 or 13x13. Each cell in this grid takes on the responsibility of detecting objects.

b) Bounding Box Predictions: YOLO predicts multiple bounding boxes within each grid cell. These boxes serve as hypotheses regarding potential object locations. For each bounding box, the algorithm predicts essential parameters, including the coordinates of the box's center, its width and height, and a confidence score denoting the model's certainty that the box contains an object.

c) Class Prediction: YOLO simultaneously provides predictions for the probability of the detected object longing for various classes. This facilitates multi-class object detection, allowing users to identify and classify different objects within the image.

d) Confidence thresholding and Non-Maximum Suppression: YOLO applies a confidence threshold to filter out less certain detections after making predictions. Subsequently, it employs Non-Maximum Suppression (NMS) to refine the results. NMS ensures that only the most confident and non-overlapping bounding boxes are retained, eliminating redundancy in predictions.

e) Single Forward Pass: Unlike other object detection algorithms that perform multiple passes over an image, YOLO processes the entire image in a single forward pass. Thus, it enhances speed and makes YOLO particularly suitable for real-time applications.

B. Retinanet Overview

RetinaNet is a state-of-the-art object detection model that has gained prominence in the field of computer vision. Noteworthy for its ability to strike a balance between accuracy and efficiency. The working of RetinaNet includes the following steps:

a) Feature Pyramid Network (FPN): RetinaNet employs a Feature Pyramid Network to extract hierarchical features from an image [7]. This is crucial for detecting objects of various sizes, ensuring that fine details and coarse structures are considered during detection.

b) Anchor Boxes: Similar to other object detection models like YOLO, RetinaNet utilizes anchor boxes. These anchor boxes serve as reference points across the image, spanning different scales and aspect ratios. They act as hypotheses for potential object locations.

c) Object Classification: For each anchor box, RetinaNet predicts the likelihood of it containing an object and assigns a specific class to the object. The classification is performed through a dedicated sub-network, allowing RetinaNet to handle multi-class detection tasks.

d) Bounding Box Regression: In parallel with classification, RetinaNet engages in bounding box regression [8]. The model refines the initially proposed anchor boxes to better align with the true boundaries of the detected objects. This step enhances the precision of object localization.
e) Focal Loss: RetinaNet introduces a novel loss function called Focal Loss [9]. This addresses the issue of class imbalance inherent in object detection datasets. Focal Loss dynamically adjusts the weights assigned to different examples during training, prioritizing challenging cases over well-classified ones, which represents quantifiable advancement in identifying underrepresented classes with enhanced precision and recall as compared to traditional approaches.
f) Single Shot Processing: RetinaNet uses a single-shot processing approach to evaluate the entire image in a single forward pass. This streamlined method improves the model's efficiency, making it adept at real-time object detection applications.
The integration of FPN, anchor boxes, and the introduction of Focal Loss collectively contribute to its success in achieving accurate and efficient object detection results.

C. *Efficientdet Overview*

EfficientDet is acclaimed for its efficiency in balancing accuracy and computational resources [10]. The working of EfficientDet is mentioned in the following steps:
a) Efficient Backbone Architecture: At the core of EfficientDet is a highly efficient backbone architecture. Unlike traditional models that often employ resource-intensive backbones, EfficientDet utilizes compound scaling to achieve a favourable balance between model size and accuracy. This enables the model to operate efficiently across a range of computational resources.
b) BiFPN (Bidirectional Feature Pyramid Network): EfficientDet employs a Bidirectional Feature Pyramid Network to capture multi-scale features effectively. [11] This innovative network design facilitates information exchange across different scales, enhancing the model's ability to detect objects of various sizes within an image.
c) Anchor Box Mechanism: Similar to other object detection frameworks, EfficientDet leverages anchor boxes [12]. These anchor boxes, strategically positioned across the image, serve as reference points for potential object locations. The model predicts the presence of objects within each anchor box and refines their positions through bounding box regression.
d) Object Classification: EfficientDet incorporates a robust mechanism for object classification. Using a dedicated sub-network, the model assigns specific class labels to detected objects. This allows EfficientDet to handle tasks involving multiple object classes.
e) EfficientDet Loss Function: The model employs a well-crafted loss function that combines both classification and regression objectives [13]. This loss function ensures that the model accurately classifies objects and precisely localizes them within the image.
f) Compound Scaling: One of the distinctive features of EfficientDet is its use of compound scaling to optimize model parameters. This technique scales the model's depth, width, and resolution in a balanced manner, resulting in improved accuracy without excessive computational demands.
g) Efficient Single Shot Processing: Following the trend of single-shot object detection models, EfficientDet processes the entire image in a single pass [14]. This efficient single-shot processing contributes to the model's real-time object detection capabilities.

VI. MATERIALS AND FRAMEWORK

A. *Dataset*

We used Roboflow, a flexible data management tool, to carefully select a portion of the large COCO dataset. We started with 7,500 photographs in three categories but had to narrow them down to a training set of 100 images per category due to practical limitations. [15] Each category is then evaluated using six test images. The dataset was thoroughly preprocessed before training, with all photos resized to a standard 416x416 pixel size and rotation augmentations applied to increase diversity within the training set. The enormous diversity of the COCO dataset and Roboflow's effective data processing powers were combined to create a training experience that was both nuanced and highly optimized.

B. *Tools and Techniques*

We used a number of essential tools and libraries in our Google-Colab project, as listed below:
a.TensorFlow We use this open-source machine learning framework [16] to develop and train our neural network models for their scalability and adaptability, making it a popular platform for various machine learning applications.
b.PyTorch: We used PyTorch because of its dynamic computational graph [17], which is especially useful for deep learning research and testing. Its simple layout and effortless operation made integration into our project a breeze.
Pandas: Pandas is a flexible Python data manipulation library. It was used to handle and preprocess data efficiently [18]. We could efficiently organize and handle our dataset thanks to its Data-Frame format, which also helped expedite the data preparation process.
c.NumPy: Python numerical calculations can be performed with the help of the NumPy library. Since NumPy offers a strong foundation for various mathematical and statistical operations crucial to machine learning processes [19], we rely on it for our array-based computations.
d. OpenCV: For image processing jobs, an Open Source Computer vision library was essential [20]. For tasks like loading images, resizing them, and performing different computer vision operations, we used OpenCV. It was an essential tool for handling and modifying image data due to its extensive feature set.
Overall, TensorFlow, PyTorch, Pandas, NumPy, OpenCV, and Google Colab helped us create a strong ecosystem that made building and training our machine-learning models easier. Every tool had a unique function that enhanced the effectiveness and success of our endeavour.

C. Preprocessing

The dataset was already available in different formats on Roboflow. The images were resized to 416 x 416, and rotational augmentations were applied. After image preprocessing, the dataset directory structure was made to match the input format of that model.

V. EXPERIMENT

The Ultralytics YOLOv8 represents a leading-edge, state-of-the-art (SOTA) model that enhances upon the accomplishments of its predecessors, incorporating novel features and enhancements to augment overall performance and adaptability. YOLOv8 is meticulously crafted to be swift, precise, and user-friendly. It is an outstanding option for diverse applications such as object detection and tracking, instance segmentation, image classification and pose estimation tasks. As compared to YOLOv8's predecessor, YOLOv5, YOLOv8 comes with a new anchor-free detection system, changes to the convolutional blocks used in the model, and mosaic augmentation applied during training, turned off before the last 10 epochs. It stores the dataset path and classes in the YAML file and was trained by modifying the pre-trained weights provided by Ultralytics. The dataset was converted to YOLOv8 format by Roboflow Universe.

Fizyr solutions are employed for custom dataset training with RetinaNet. The RetinaNet model undergoes training using pre-trained weights from the COCO dataset and a ResNet50 backbone, comprising a total of 12,863,295 trainable parameters. Each annotation is represented on a single line using a CSV file to pass data. When images contain multiple bounding boxes, each box is assigned its own row in the CSV file. Fizyr's keras-retinanet training procedure employs models specifically designed for training purposes, with streamlined versions compared to the inference model featuring only the essential layers for regression and classification values. It is imperative to convert the trained model into an inference model to conduct inference on a model, which is necessary for object detection in an image.

In the case of EfficientDet, another EfficientDet Pytorch repository has been utilized. This is a Pytorch re-implementation of the official EfficientDet, showcasing state-of-the-art (SOTA) performance in real-time scenarios. The model encompasses multiple coefficients, ranging from D0 to D8, where D8 exhibits the highest mean average precision (mAP). The model requires input in a dataset format akin to Microsoft COCO, as the repository specifies. A YAML-formatted configuration file is included, enabling the manual configuration of project-specific parameters. During this model training, the backbones are frozen, and the training parameters are as follows:

- Epochs: 50
- Patience: 15
- Learning Rate: 0.001
- Batch size: 16

The dataset is partitioned into training and testing sets, and the model undergoes updates across multiple epochs, contingent upon the specified tolerance for optimization (TOL). Convergence is deemed achieved, and training concludes when there is no significant improvement in the loss or score by at least TOL for consecutive iterations as defined.

IV. RESULTS AND DISCUSSION

This investigation involves an assessment of the performance of three object detection models – YOLOv8, Efficient- Det, and RetinaNet, employing precision, recall, and mean Average Precision (mAP) as pivotal metrics. Figure 1 represents the precision of Yolov8, RetinaNe and EfficiantDet algorithms.

FIGURE 1: precision of Yolov8, RetinaNet and EfficiantDet algorithms

TABLE I
PERFORMANCE OF YOLOV8, RETINANET AND EFFICIANTDET ALGORITHMS

Metric	YOLOv8	RetinaNet	EfficientDet
Accuracy	0.71 (highest)	0.67	0.55
Inference Time	1.0 ms (fastest)	1.2362 s (slowest)	1.0 s
mAP	0.71 (highest)	0.67	0.55

Table 1 shows the performance of all three algorithms based on their accuracy, Inference time and mAp value. As mentioned in Table 1, YOLOv8 showed the most elevated accuracy among the models, with a score of 0.71, demonstrating areas of strength to detect positive occurrences accurately.

RetinaNet exhibited an estimable accuracy of 0.67, displaying its precision in accurately identifying positive cases. EfficientDet had a precision value of 0.55, respectively, which was slightly lower, but it still performed well enough to correctly identify true positives.

IYOLOv8 has the highest speed (inference time) of 1.0ms, mirroring its viability in catching significant positive occurrences in a single detection.

RetinaNet detects objects in multiple stages, so it has a high inference time of around 1.2362s.

EfficientDet was followed intimately with an inference speed of 1s, showing a comparable ability to detect positive instances. With a mAP score of 0.71, YOLOv8 stands out as a strong contender for object detection tasks due to its overall proficiency in precision and recall. RetinaNet showed a competitive mAP value of 0.67, which is a reasonable performance. EfficientDet had a respectable level of accuracy with a mAP of 0.55, but lagged slightly behind the other models. Figure 2 exhibits the mAP values

of all three algorithms.

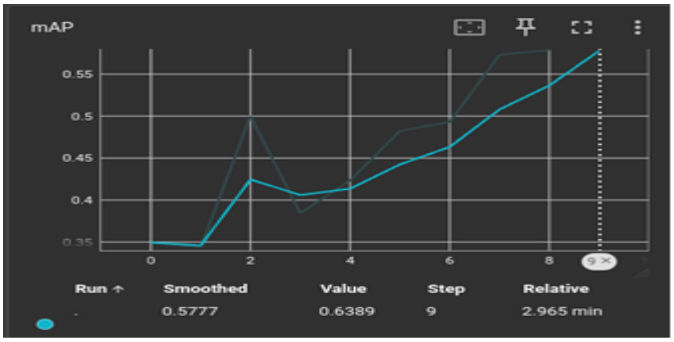

FIGURE 2: mAP for Yolov8, RetinaNet and EfficientDet Algorithms.

VII. CONCLUSION

The comparative examination of performance metrics, mAP assessment, and inference times across the three object detection models indicates that YOLOv8 achieves the highest levels of accuracy, recall, and mAP scores, positioning it as a robust candidate for object detection tasks. Nevertheless, RetinaNet and EfficientDet also demonstrate competitive performance, offering potential advantages in certain use cases or where specific computational requirements are prioritized. YOLOv8 proves effective for real-time object detection, particularly for small objects, excelling in scenarios emphasizing high accuracy and recall. Conversely, RetinaNet's utilization of focal loss enhances its capability to handle class imbalance, and EfficientDet's scalability and efficiency make it an attractive choice for re-source-constrained environments. These insights provide valuable guidance for researchers and practitioners in selecting the most appropriate object detection model tailored to their specific needs and constraints.

ACKNOWLEDGMENT

The authors would like to express their gratitude to "the Software Engineering Department of NED University of Engineering and Technology" for their invaluable support and resources throughout this research.

FUNDING STATEMENT

The author(s) received no specific funding for this study.

CONFLICTS OF INTEREST

The authors declare they have no conflicts of interest to report regarding the present study.

REFERENCES

[1] Amin, M.A. and K. Khaled, Copper corrosion inhibition in O2- AI Pioneer, "Medium," 28 June 2023. [Online]. Available: https://medium.com/@tejasdalvi927/object-detection-with-yolo-and-opencv-a-practical-guide-cf7773481d11. [Accessed 9 January 2024].

[2] N. Yadav and B. Utkarsh, "Comparative Study of Object Detection Algorithms," International Research Journal of Engineering and Technology (IRJET) , vol. 4, no. 11, pp. 586-591, 2017.

[3] R. Padilla, R. L. Passos, T. L.B. Dias, S. L. Netto and S. A.B. da Silva, "A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit," Electrical Engineering Program/Alberto Luiz Coimbra Institute for Post-Graduation and Research in Engineering, 2021.

[4] M. Haris and A. Glowacz, "Road Object Detection: A Comparative Study of Deep Learning-Based Algorithms," Electronics 2021, 2021.

[5] S. Kuutti, R. Bowden, Y. Jin, P. Barber and S. Fallah, " A Survey of Deep Learning Applications to Autonomous Vehicle Control," IEEE Trans. Intell. Transp. Syst, p. 712–733, 2020.

[6] R. D. J. D. T. M. J. Girshick, "Rich feature hierarchies for accurate object detection and semantic segmentation.," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 38, no. 1, pp. 142-158, 2014.

[7] X. Yang, W. Wang, J. Wu, C. Ding and S. Ma, "MLA-Net: Feature Pyramid Network with Multi-Level Local Attention for Object Detection," Advanced Machine Learning Methods for Image Processing, Perception and Understanding, 2022.

[8 S.-h. Tsang, "Medium," 14 January 2019. [Online]. Available: https://towardsdatascience.com/review-retinanet-focal-loss-object- detection-38fba6afabe4. [Accessed 11 January 2024].

[9] A. Kirouane, "Linkedin," 4 February 2023. [Online]. Available: https://www.linkedin.com/pulse/retinanet-focal-loss-object-detection- ayoub-kirouane/. [Accessed 11 January 2024].

[10] M. Tan, R. Pang and Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition , 2020.

[11] L. Fu, W.-b. Gu, W. Li, L. Chen, Y.-b. Ai and H.-l. Wang, "Bidirectional parallel multi-branch convolution feature pyramid network for target detection in aerial images of swarm UAVs," Defence Technology, vol. 17, no. 4, pp. 1531-1541, 2021.

[12] J. Li, "Medium," 3 November 2020. [Online]. Available: https://blog.ml6.eu/retraining-efficientdet-for-high-accuracy-object- detection-961e906cae8. [Accessed 11 January 2024].

[13] M. Tan, R. Pang and Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection," in IEEE, 2020.

[14] R. Kundu, "V7," 17 January 2023. [Online]. Available:https://www.v7labs.com/blog/yolo-object-detection. [Accessed 11 January 2024].

[15] E. Padron, "Medium," 30 July 2023. [Online]. Available: https://fulldataalchemist.medium.com/building-your-own-real-time- Object-detection-app-roboflow-yolov8-and-streamlit-part-1-f577cf0aa6e5. [Accessed 9 January 2024].

[16] S. Yegulalp, "InfoWorld," 5 January 2024. [Online]. Available: https://www.infoworld.com/article/3278008/what-is-tensorflow-the- machine-learning-library-explained.html. [Accessed 9 January 2024].

[17] V. K. Solegaonkar, "towardsdatascience," 20 September 2019. [On- line]. Available: https://towardsdatascience.com/introduction-to-py-torch- 13189fb30cb3. [Accessed 9 January 2024].

[18] C. Team, "CodeAcademy," [Online]. Available: https://www.codecademy.com/article/introduction-to-numpy-and-pandas. [Accessed 9 Januray 2024].

[19] C. Team. [Online]. Available: https://www.codecademy.com/article/introduction- to-numpy-and-pandas. [Accessed 9 January 2024].

[20] Boesch, "viso.ai," [Online]. Available: https://viso.ai/computer-vision/opencv/. [Accessed 9 January 2024].Standard Y10.5-1968.