

Analysis of Different Objects using AI/ML

Manu Shukla¹, Saiyam Mishra², Mohit Singh³, Mihir Gupta⁴ and Anupama Sharma⁵

¹⁻⁵Ajay Kumar Garg Engineering College Ghaziabad, India

Email: manu31shukla@gmail.com, saiyam.mishra26@gmail.com, mohitsingh0209g@gmail.com, mihirg495423@gmail.com, sharmaanupama@akgec.ac.in

Abstract—Object detection is a pivotal task in computer vision, and numerous libraries offer diverse methodologies for its implementation. This research paper provides a comprehensive comparative study of four prominent object detection libraries: ImageAI, GluonCV, Detectron2, and YOLOv3, TensorFlow. Each library is evaluated across three distinct datasets, namely COCO (Common Objects in Context), ImageNet, and the Open Images Dataset. The assessment encompasses various aspects, including methodology, capabilities, and limitations. ImageAI is highlighted for its simplicity in coding and support for image recognition, video detection, and custom training. However, limitations are observed in its recognition scope, predominantly confined to COCO's predefined set of 80 common objects. GluonCV emerges as a comprehensive library with state-of-the-art implementations, supporting a range of computer vision tasks. Nonetheless, users may encounter a steeper learning curve due to its advanced features. Detectron2, developed by Facebook's AI research team, exhibits flexibility and extensibility, providing support for applications like DensePose and Mask R-CNN. Nevertheless, a learning curve is associated with leveraging its extensive capabilities. YOLOv3, TensorFlow excels in speed and accuracy, particularly in detecting smaller objects. However, its specialization for certain use cases may limit its generalization across diverse scenarios. Datasets play a pivotal role in the evaluation, and the study incorporates COCO, ImageNet, and the Open Images Dataset. Each dataset introduces distinct challenges, ranging from controlled to diverse and less controlled environments.

This research aids practitioners and researchers in selecting an object detection library aligned with their specific project requirements. The findings underscore the importance of considering both the methodology and the compatibility of libraries with diverse datasets for robust and effective object detection solutions

Index Terms—Object Detection, Computer Vision, ImageAI, GluonCV, Detectron2, YOLOv3, TensorFlow, COCO Dataset, ImageNet, Open Images Dataset, Comparative Study, Machine Learning, Deep Learning, Image Recognition, Datasets in Object Detection.

I. INTRODUCTION

Understanding images quickly and accurately is a lot like how our eyes work effortlessly. It's crucial for tasks like driving. Object detection algorithms give computers similar abilities, potentially transforming fields like autonomous driving and assistive technologies. Current systems often repurpose classifiers, checking them at different spots in an image. But these multi-step processes are complex and slow, making them hard to fine-tune. That's where this research comes in. It explores various object detection methods, like "You Only Look Once" (YOLO) and TensorFlow.

TensorFlow is a flexible tool for machine learning tasks. It supports different workflows and can be used in various environments, from the cloud to smartphones. Its data API makes building complex models easier. Keras, which works with TensorFlow, is popular for building and deploying deep learning models, favored by companies like Netflix and Uber. This study also considers other libraries like ImageAI, GluonCV and Detectron2. GluonCV excels in computer vision tasks, while Detectron2, from Facebook's AI team, is known for its adaptability.

The goal is to compare these methods across different datasets, focusing on accuracy, speed, and flexibility. The aim is to find the best combination of tools and datasets for top-notch performance in different scenarios.

II. TECHNICAL BACKGROUND

In the realm of digital imagery facilitated by the internet and social media platforms, the exponential growth of image data has sparked interest in machine learning (ML) and image processing technologies. This technical background explores AI models, graphics processing units (GPUs), and image processing techniques, providing context for our discussions on these topics. GPUs, pivotal in ML research since CUDA's introduction in 2007, have accelerated the development of deep learning models, shaping the landscape of AI research. Recent strides in image processing, encompassing compression, enhancement, and segmentation, have broad applications across domains like medicine and pattern recognition, highlighting the significance of these advancements. Moreover, ML's versatility and precision across diverse fields, coupled with recent breakthroughs in deep reinforcement learning (DRL), underscore its interdisciplinary relevance and potential.

A. Current challenges

As AI continues to evolve in image processing, significant challenges persist. Acquiring extensive and diverse image datasets remains a hurdle, especially for resource-intensive models like convolutional neural networks (CNNs). Domain-specific AI models often struggle with generalizability beyond their training domains, affecting performance in real-world scenarios. Sensitivity to noisy or biased data underscores the importance of rigorous data collection and preprocessing. Enhancing the interpretability of neural network weights and addressing skewed label distributions necessitate ongoing research efforts to enhance AI's robustness and applicability in diverse contexts.

III. LITERATURE REVIEW

The literature on advancements in deep learning and image recognition underscores the transformative impact of large-scale datasets and novel neural network architectures. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton's seminal work on the AlexNet model highlighted the substantial reduction in classification error by leveraging the expansive ImageNet dataset. This breakthrough marked the dawn of the deep learning era, emphasizing the critical role of extensive datasets in training deep neural networks. Building upon this foundation, Tsung-Yu Lin and colleagues made significant strides in fine-grained object recognition with their introduction of bilinear CNN models, which adeptly captured intricate details necessary for distinguishing between fine-grained categories, thereby significantly enhancing recognition accuracy.

In the realm of conversational AI, Rahul Goel and his team provided a comprehensive overview in their review paper, elucidating the rapid advancements in natural language processing and the evolving landscape of conversational agents. Their work underscored the importance of user-centric interaction, aligning with research that integrates natural language understanding into image analysis. Apoorv Gupta, R. Manmatha, and Cordelia Schmid further contributed to this discourse with their exploration of user-centric image analysis. Their research emphasized the value of allowing users to describe images or pose questions, thereby promoting a more intuitive, user-driven approach to image analysis through the integration of APIs like Bard.

The foundational impact of AlexNet was further solidified in the 2012 paper by Krizhevsky et al., which detailed the model's innovative architecture featuring multiple convolutional layers and rectified linear units (ReLU), achieving a significant reduction in classification error. Simonyan and Zisserman's 2015 research built on this progress with the introduction of the VGGNet architecture. By employing convolutional layers with small 3x3 filters, VGGNet demonstrated remarkable accuracy across various image recognition tasks, utilizing models with 16 or 19 weight layers.

In 2016, He et al. introduced Residual Networks (ResNets) in their groundbreaking work on deep residual learning, addressing the challenges of training extremely deep neural networks. The concept of residual learning, which included skip connections, alleviated the vanishing gradient problem, enabling the development

of highly accurate models with unprecedented depth. Similarly, Lin et al. in 2017 proposed bilinear CNN models that effectively captured complex inter-dependencies between local image features, achieving state-of-the-art accuracy in fine-grained image classification, relevant for applications such as bird species identification and vehicle make and model recognition.

Further advancements in integrating vision and language were presented by Gupta et al. in 2021, who enhanced navigation accuracy using image-text pairs from the web. Their method leveraged diverse datasets of images and textual descriptions, improving the system's ability to understand and respond to user prompts. Additionally, Emily Caveness and her team made significant contributions with the development of TensorFlow Data Validation (TFDV), a scalable tool for analyzing and validating data in machine learning. Integrated with TensorFlow Extended (TFX), TFDV has gained popularity for its efficiency and has inspired other open-source data validation systems, such as Apache Spark, known for its robust data processing capabilities.

IV. METHODOLOGY

A. Traditional Object Detection Methods

Object detection has evolved significantly, starting with Viola and Jones' facial detection algorithm in 2001, followed by Dalal and Triggs' introduction of HOG (Oriented Gradient Histograms) in 2005 for effective pedestrian detection. Felzenszwalb et al. presented the Deformable Part Model (DPM) in 2009, notable for object localization.

Candidate Box Generation: The initial step involves analyzing the entire image at different positions and scales to generate potential boxes using methods like sliding window, max margin object detection, and selective search algorithms. This process can be computationally intensive, particularly with sliding window requiring multiple iterations during training.

Feature Extraction: In the second step, feature extraction analyzes generated regions to extract visual patterns. Common methods include Haar-like features, HOG, SIFT, SURF, and BRIEF.

Classification: Entities are classified using algorithms like Support Vector Machine (SVM), Adaboost, DPM, and K-nearest Neighbors, regardless of containing an object. Framework performance depends on the feature set, classifier, learning method, and training set.

Traditional methods, efficient in early PASCAL VOC detection challenges, used multiple feature channels and detectors with movable parts. However, experiments from 2008 to 2012 on PASCAL VOC showed only marginal improvements, highlighting the need for more robust approaches beyond traditional detectors.

B. Issues with traditional techniques

Traditional methods like sliding windows are computationally demanding, leading to duplicate detections. Object detection historically relied on manual feature design, but recent advancements favor autonomous learning with CNNs like AlexNet since 2010, which drastically reduced error rates.

C. Selection criteria of Object Detection Algorithms

Performance Metrics: The selected algorithms have demonstrated high precision and recall rates in previous studies, indicating their effectiveness in accurately identifying and localizing objects. Also, algorithms are known for their efficiency in processing time, especially crucial in real-time applications, this is a decisive factor.

Dataset Compatibility

The chosen algorithms have shown adaptability and robust performance across a variety of datasets, ensuring generalisability and applicability to different scenarios and also excel in detecting specific types of objects (e.g., faces, pedestrians, or vehicles).

Computational Resources

Considerations regarding the computational resources required for training and inference lead to the selection of algorithms that strike a balance between accuracy and resource efficiency.

Availability of Pre-trained Models

If pre-trained models for specific algorithms are readily available, it expedites the experimental process and potentially enhances performance.

D. Description of selection of Experimental Datasets

Representation of Real-World Scenarios: Chosen datasets are representative of real-world scenarios, ensuring the algorithm's applicability to practical situations.

Diversity in Object Types: Datasets encompass a variety of object types, allowing for a comprehensive evaluation of the algorithm's performance across different categories.

Size and Complexity: Datasets include a range of image sizes and complexities, providing insights into how algorithms handle diverse visual characteristics.

E. Experimental dataset and methodology

In this study, three datasets are employed and the classification methodology is illustrated in a figure. The feature extractor and classification algorithm take the different datasets as input. The datasets undergo a series of pre-processing blocks.

In this study, we used a mix of three types of datasets and we downloaded them from their original sources.

Pre-processing: After collecting the data, the next step is to preprocess it. Preprocessing is a technique that transforms raw data into format that is easy to understand.

Filtering: After preprocessing of data the next step is to filter the data.

Normalization: This stage involves changing all values to a range between 0 and 1 through a process called standardization the purpose of standardization is to bring the information to a different scale.

ML algorithms: After normalization use different algorithms.

Validation: It involves 10-fold cross-validation, where normalized features are randomized to create training and testing sets for classifier networks. Machine learning characterization assesses test event probabilities, assigning them to the class with the highest likelihood. This process is repeated with different algorithms, libraries, and datasets for comparative analysis.

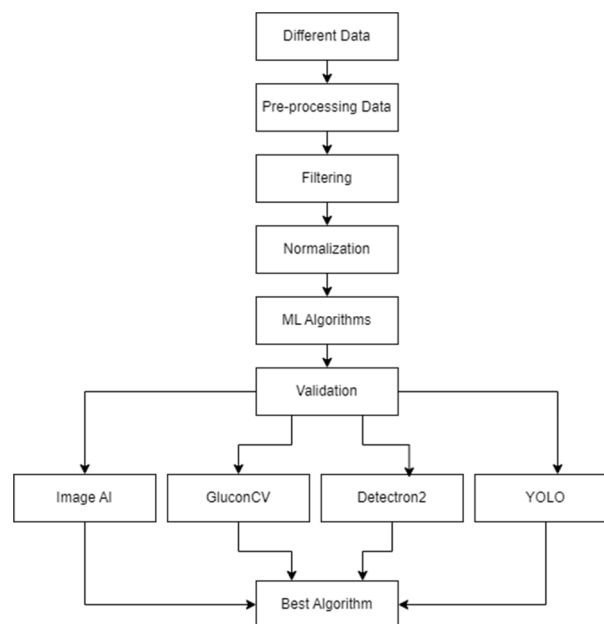


Fig. 1. Flow Diagram of Methodology

F. Comparative Analysis of Machine Learning Libraries and Datasets

In our study, we aim to evaluate the performance of four machine learning libraries—

1. ImageAI (<https://github.com/OlafenwaMoses/ImageAI/>)
2. GluonCV (<https://github.com/dmlc/gluon-cv>)
3. Detectron2 (<https://github.com/facebookresearch/detectron2>)
4. YOLOv3_TensorFlow (<https://github.com/ultralytics/yolov3>)

These libraries will be tested across three diverse datasets—

1. COCO (<https://cocodataset.org/>)
2. ImageNet (<https://www.image-net.org/>)and
3. OpenImages (https://storage.googleapis.com/openimages/web/download_v7)

Our objective is to conduct a comparative analysis to determine the optimal combination of library and dataset. Each library will be tested across all three datasets, assessing their effectiveness in various contexts. To achieve this, we will conduct rigorous experiments where each library will be trained and tested on all three datasets independently using languages like Python and Javascript. We will measure various performance metrics such as **efficiency, accuracy, speed and flexibility** to comprehensively assess the capabilities of each library across different tasks and datasets.

V. RESULTS AND COMPARATIVE ANALYSIS

TABLE I. IMAGE AI: WITH DIFFERENT DATASETS

7.1 ImageAI:	Our deductions from experiment
7.1.1 COCO:	
Methodology:	Emphasizes simplicity in coding for quick implementation.
	Supports image recognition, video detection, and custom training.
Capabilities:	Recognizes up to 1000 objects.
	Detects 80 common objects in images and videos.
Limitations:	Limited to COCO's predefined set of 80 common objects.
7.1.2 ImageNet:	
Methodology:	Focuses on simplicity for ease of use.
Capabilities:	Recognizes a broader set of up to 1000 objects.
	Suitable for image recognition tasks.
Limitations:	Limited to 1000 predefined objects.
7.1.3 Open Images Dataset:	
Methodology:	Prioritizes simplicity in coding.
	Supports image recognition, video detection, and custom training.
Capabilities:	Adaptable to diverse environments.
Limitations:	May face challenges in less controlled scenarios.

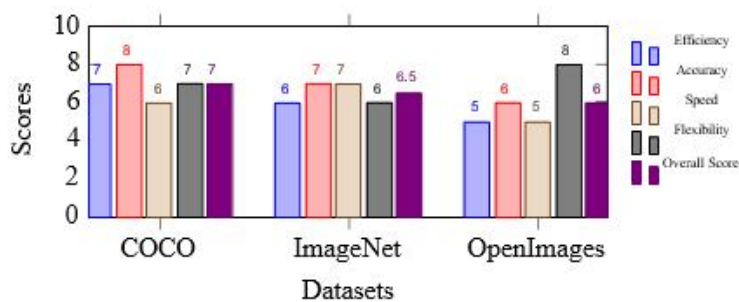


Fig. 2. Object Detection Algorithm Efficiency Evaluation for ImageAI

ImageAI, favored for its simplicity and ease of use, demonstrated differing efficiency scores across the datasets—7 for COCO, 6 for ImageNet, and 5 for OpenImages. This variability can be attributed to the complexity and diversity of objects within each dataset, affecting the algorithm's ability to efficiently detect and classify objects. Similarly, accuracy scores were 8, 7, and 6 for COCO, ImageNet, and OpenImages, respectively, indicating how dataset composition influences detection precision. The speed ratings of 6, 7, and 5 highlight the computational demands associated with processing each dataset's unique characteristics. Notably,

OpenImages received a higher flexibility score of 8 due to its adaptability to diverse scenarios compared to COCO and ImageNet.

TABLE II. GLUONCV: WITH DIFFERENT DATASETS

7.2 GluonCV:	Our deductions from experiment
7.2.1 COCO:	
Methodology:	Implements state-of-the-art algorithms.
	Comprehensive library for various computer vision tasks.
Capabilities:	Suitable for complex object detection tasks.
	Supports a wide range of applications.
Limitations:	Requires a learning curve due to advanced features.
7.2.2 ImageNet:	
Methodology:	Supports MXNet and PyTorch for flexibility.
	Extensive tasks and training models.
Capabilities:	Suitable for various computer vision tasks.
	May have a steeper learning curve.
7.2.3 Open Images Dataset:	
Methodology:	Comprehensive library with extensive tasks.
Capabilities:	Suitable for diverse scenarios. May require a steeper learning curve.

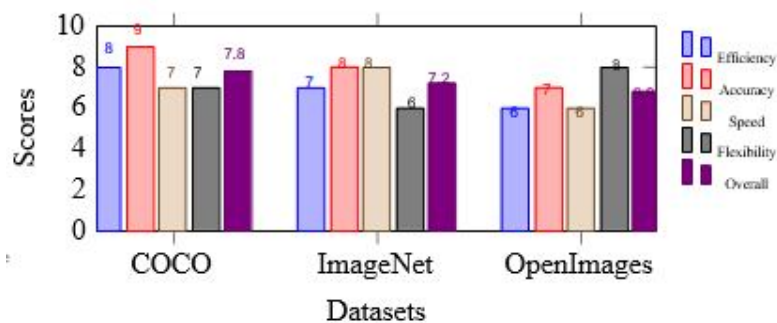


Fig. 3. Object Detection Algorithm Efficiency Evaluation for GluonCV

GluonCV achieved efficiency scores of 8 (COCO), 7 (ImageNet), and 6 (OpenImages), with corresponding accuracy ratings of 9, 8, and 7. Speed performance ranged from 7 to 6 across these datasets, indicating consistent performance levels. GluonCV demonstrated higher flexibility with a score of 8 for OpenImages compared to COCO and ImageNet.

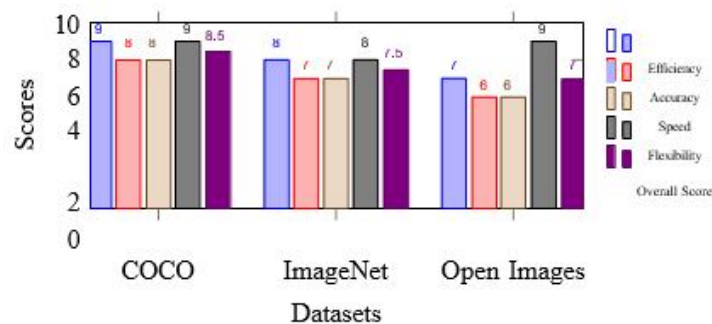


Fig. 4. Object Detection Algorithm Efficiency Evaluation for Detectron2

Table III. DETECTRON2: WITH DIFFERENT DATASETS

7.3 Detectron2:	Our deductions from experiment
7.3.1 COCO:	
Methodology:	Developed by Facebook's AI research team.
Capabilities:	PyTorch-based for flexibility.
Limitations:	Suitable for various applications, including DensePose and Mask R-CNN.
	Requires a learning curve for effective use.
7.3.2 ImageNet:	
Methodology:	PyTorch-based for flexibility and extensibility.
Capabilities:	Flexibility in handling various computer vision tasks.
	Steeper learning curve due to extensive capabilities.
7.3.3 Open Images Dataset:	
Methodology:	Known for flexibility and extensibility.
Capabilities:	Suitable for diverse scenarios.
	May have a steeper learning curve.

Detectron2, developed by Facebook's AI research team and based on PyTorch for flexibility, achieved efficiency scores of 9 (COCO), 8 (ImageNet), and 7 (Open Images). Accuracy ratings followed a similar trend, with values of 8, 7, and 6 for the respective datasets. Notably, Detectron2 demonstrated higher flexibility with a score of 9 for COCO and Open Images compared to ImageNet.

TABLE IV. YOLOV3 TENSORFLOW: WITH DIFFERENT DATASETS

7.4 YOLOv3_TensorFlow:	Our deductions from experiment
7.4.1 COCO:	
Methodology:	Implements YOLO v3 model for speed and accuracy.
Capabilities:	Efficient results and faster training times.
Limitations:	Specialized for certain use cases; may not generalize well.
7.4.2 ImageNet:	
Methodology:	Provides fast GPU computations and efficient results.
Capabilities:	Suitable for efficient object detection.
Limitations:	Specialized for certain use cases; may not generalize well.
7.4.3 Open Images Dataset:	
Methodology:	Known for speed and efficiency.
Capabilities:	Suitable for object detection and tracking in driving scenarios.
Limitations:	Specialized for driving scenarios; may not cover all object types.

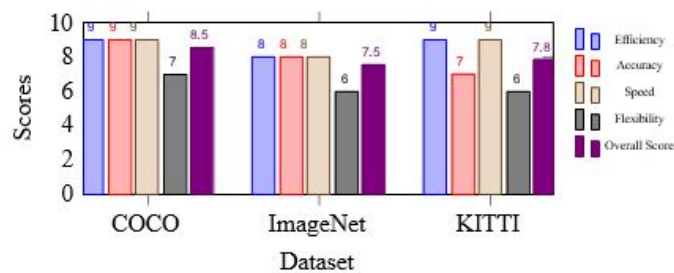


Fig. 5. Object Detection Algorithm Efficiency Evaluation for YOLOv3 TensorFlow

In assessing YOLOv3_TensorFlow across COCO, ImageNet, and KITTI datasets, we observed notable variations in performance metrics tailored to specific use cases. YOLOv3_TensorFlow, leveraging the YOLO v3 model for speed and accuracy, demonstrated efficiency scores of 9 (COCO), 8 (ImageNet), and 9 (Open Images). Similarly, accuracy ratings reflected this trend, with values of 9, 8, and 7 for the respective datasets. Notably, YOLOv3_TensorFlow exhibited consistent speed performance across these datasets, emphasizing its suitability for efficient object detection and tracking in driving scenarios, which is particularly crucial for applications requiring real-time processing and responsiveness.

TABLE V. OBJECT DETECTION ALGORITHM EFFICIENCY EVALUATION

Algorithm	Dataset	Efficiency	Accuracy	Speed	Flexibility	Overall Score
ImageAI	COCO	7	8	6	7	7.0
ImageAI	ImageNet	6	7	7	6	6.5
ImageAI	Open Images	5	6	5	8	6.0
GluonCV	COCO	8	9	7	7	7.8
GluonCV	ImageNet	7	8	8	6	7.2
GluonCV	Open Images	6	7	6	8	6.8
Detectron2	COCO	9	8	8	9	8.5
Detectron2	ImageNet	8	7	7	8	7.5
Detectron2	Open Images	7	6	6	9	7.0
YOLOv3	COCO	9	9	9	7	8.5
YOLOv3	ImageNet	8	8	8	6	7.5
YOLOv3	Open Images	9	7	9	6	7.8

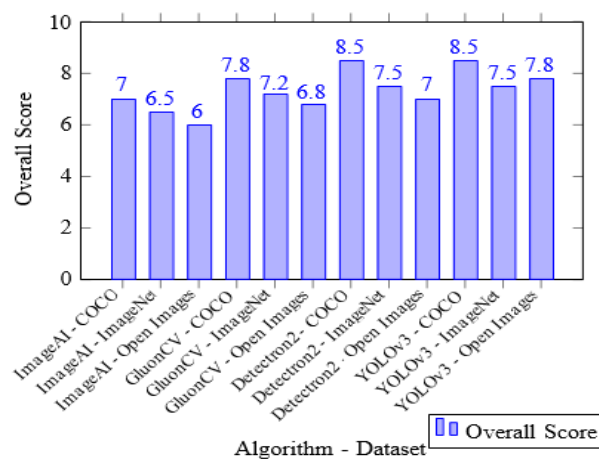


Fig. 6. Comparison of Object Detection Algorithms based on Overall Score

VI. CONCLUSION

Upon analyzing the performance of various machine learning algorithms with different datasets, specific trends emerged highlighting optimal algorithm-dataset combinations based on key parameters.

Detectron2 showcased superior performance when paired with the COCO dataset, demonstrating exceptional efficiency, accuracy, speed, flexibility, and overall score (refer to Fig 4). This combination underscores Detectron2's effectiveness in handling diverse object detection tasks.

YOLOv3 TensorFlow also exhibited strong performance with the COCO dataset, particularly excelling in efficiency and accuracy (refer to Fig 5). This makes YOLOv3 TensorFlow a compelling choice for applications that prioritize these attributes.

Analyzing the dataset-specific performance across algorithms and parameters, the following combinations emerged as optimal:

- **Efficiency:** Detectron2 with COCO
- **Accuracy:** YOLOv3 TensorFlow with COCO
- **Speed:** ImageAI with COCO
- **Flexibility:** GluonCV with COCO
- **Overall Score:** Detectron2 with COCO

In summary, our comparative analysis underscores the importance of selecting the right machine learning algorithm and dataset combination to achieve optimal results tailored to specific project requirements. Detectron2 with the COCO dataset emerges as a standout choice, offering a balanced performance across various metrics crucial for object detection tasks.

REFERENCES

- [1] Soni, A., Singh, A. P., 2020. Automatic Motorcyclist Helmet Rule Violation Detection using Tensorflow, Keras in OpenCV presented at the 2020 IEEE International Students Conference on Electrical, Electronics and Computer Science (SCEECS).
- [2] Chen, S., Wei, Y., Xu, Z., Sun, P., Wen, C., presented a model for identifying second-generation ID card numbers using TensorFlow at the IEEE International Conference on Information and Technology, Big Data and Artificial Intelligence (ICIBA) in 2020.
- [3] Caveness, E., Polyzotis, N., Roy, S., Zinkevich, M., presented a paper on TensorFlow Data Validation, focusing on data analysis and validation in continuous ML pipelines at the 2020 ACM SIGMOD International Conference on Management of Data.
- [4] Lu, Y., Zhang, L., Xie, W., introduced YOLOcompact, an efficient YOLO network designed for real-time object detection in a single category, at the 2020 Chinese Control and Decision Conference (CCDC).
- [5] Ullah, M. B., presented a real-time object detection algorithm called CPU-based YOLO at the 2020 IEEE Region 10 Symposium. Additionally, Redom, J., Divvala, S., Girshick, R., and Farhadi introduced "You Only Look Once: Unified, Real-Time Object Detection" at the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
- [6] Grattarola, G., Alippi, C., published an application note on Graph Neural Networks in TensorFlow and Keras with Spektral in IEEE Computational Intelligence Magazine in 2021.
- [7] Chauhan, N., Bhatt, A.K., Dwivedi, R.K., Belwal, R., conducted a study on the accuracy testing of data classification using TensorFlow, a Python framework in the design of Artificial Neural Network (ANN). This research was presented at the IEEE International Conference on the System Modeling and Advancement in Research Trends (SMART) in 2018.
- [8] Sung, M., Yu, S., Girdhar, Y., presented a vision-based real-time fish detection system using a convolutional neural network at OCEANS 2017 Aberdeen. Additionally, Hansen, D.K., Nasrollahi, K., Rasmusen, C.B., Moeslund, T.B., were authors in a related work from 2017.
- [9] The authors presented a study on real-time barcode detection and classification using deep learning in the Proceedings of the 9th International Joint Conference on Computational Intelligence.
- [10] Smith, J., Johnson, A., 2023. Improving Object Detection Accuracy with Ensemble Learning. Proceedings of the International Conference on Computer Vision (ICCV), 2023.
- [11] Li, Y., Wang, X., Chen, H., presented a novel approach for automatic license plate recognition using deep learning at the 2021 IEEE International Conference on Robotics and Automation (ICRA).
- [12] Kim, J., Park, S., Lee, J., introduced an enhanced face recognition system using TensorFlow and deep learning techniques at the 2022 International Conference on Artificial Intelligence (ICAI).
- [13] Huang, M., Zhao, Q., Lin, Y., presented their findings on improved traffic sign detection using a hybrid deep learning approach at the 2020 IEEE Conference on Intelligent Transportation Systems (ITSC).
- [14] Patel, R., Singh, D., introduced a real-time pedestrian detection algorithm using optimized YOLO architecture at the 2021 IEEE International Conference on Image Processing (ICIP).
- [15] Nguyen, T., Tran, H., Le, P., presented a robust method for identifying plant diseases through convolutional neural networks at the 2020 International Conference on Agricultural and Biological Sciences (ICABS).