Full-Stack Task Management App

Project Overview

The Full-Stack Task Management App is a responsive web application designed to allow users to create, categorize, and track tasks with authentication. It demonstrates core full-stack development concepts, including frontend design, backend API implementation, and database integration.

System Architecture

The application follows a client-server architecture with three major components:

- Frontend (React): Provides the user interface for authentication, task creation, and dashboard visualization.
- Backend (Node.js + Express.js): Implements RESTful APIs for authentication and task management.
- Database (MongoDB): Stores user profiles and task details in a NoSQL schema.

Core Features

- User authentication with JWT and password hashing.
- Task creation, updating, deletion, and categorization.
- Task progress tracking with status updates (Pending, In-Progress, Completed).
- Responsive dashboard for task visualization.

Workflow

- Users register/login using JWT-based authentication.
- Authenticated users create and manage tasks through the frontend.
- The frontend communicates with backend APIs for CRUD operations.
- MongoDB stores and retrieves user-specific tasks, which are displayed on the dashboard.

Tech Stack

- Frontend: React
- Backend: Node.js, Express.js
- Database: MongoDB
- Authentication: JWT, bcrypt

Sample APIs & Database Schema

Example APIs:

- POST /api/auth/register – Register a new user.

- POST /api/auth/login – Authenticate user and return JWT.
- POST /api/tasks – Create a new task.
- GET /api/tasks – Retrieve all tasks for the logged-in user.
- PUT /api/tasks/:id – Update a task.
- DELETE /api/tasks/:id – Delete a task.

Example MongoDB Schema:

- User: { username, email, passwordHash, role }
- Task: { title, description, category, status, dueDate, userId }

Future Enhancements

- Real-time task updates using WebSockets (Socket.io).
- Collaboration features for multiple users.
- Push/email notifications for deadlines.
- Deployment using Docker and cloud services (AWS/Heroku).
- Unit testing with Jest/Mocha.

Common Interview Questions

- How did you design the authentication system and why use JWT?
- What are the advantages of MongoDB over SQL databases for this project?
- How does the frontend communicate with the backend?
- How do you handle errors and validation in APIs?
- How would you scale this application for thousands of users?
- If you had more time, what features would you add and why?

FISHEYE HELMET OBJECT DETECTION

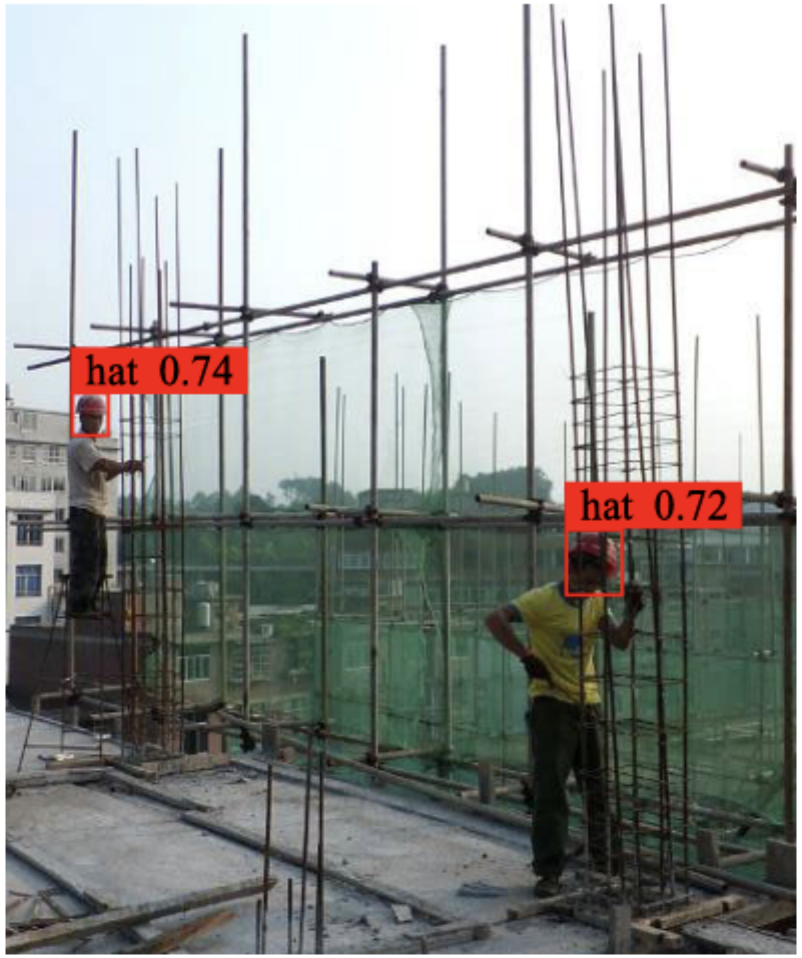
2022UCA1870- PRATHAM BHUSHAN
2022UCA3011- ASHISH GUPTA
2022UCA1832- ARYAN GADPAYLE

Under the supervision of
Dr. ANKUSH JAIN

TABLE OF CONTENT :

Introduction

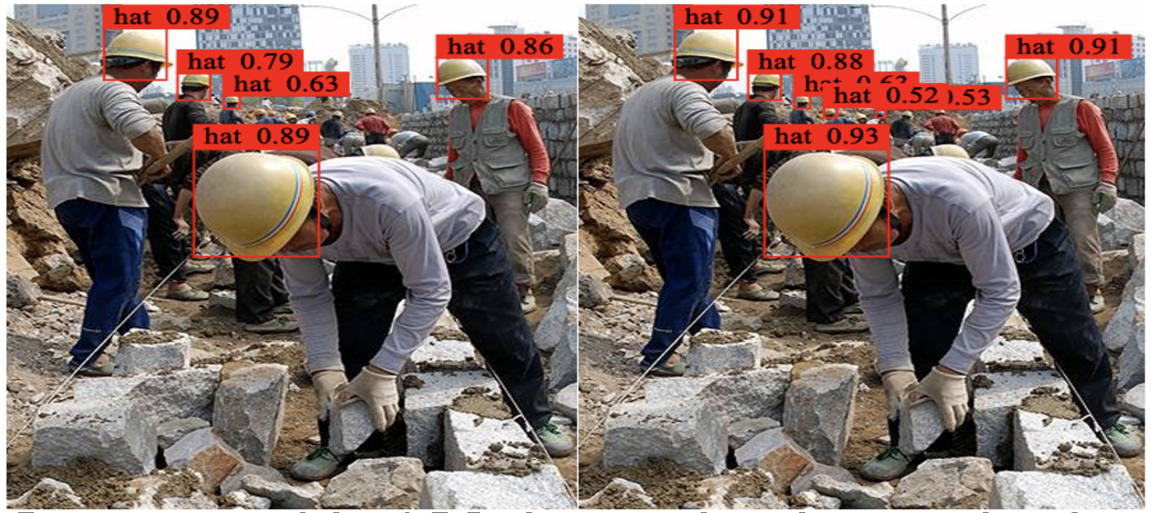
- Workplace safety is paramount, particularly in industrial environments, where head injuries pose significant risks. This project focuses on automating safety helmet monitoring using advanced computer vision technology.
- However, traditional cloud security often relies on **manual checks** , which are vulnerable to:
  - ➢ Workers safety
  - ➢ Non Adherence to safety protcol
  - ➢ Risk of life
  - ➢ Risky worspace



Introduction

To address these issues, our project integrates:

- **Yolo v5(M)** : A Yolo version 5 with lower parameters and more efficiency



- **Fisheye Lens Images**: wide 180 degree landscape (reduces no. of images)
- **Computer Vision Technology**: uses MobileNet.v3, BiCAM (channel attnetion module + spatial attention module)
- The goal is to provide a **less computational**, **efficiency driven**, and **resource-efficient** detection system for safe workers environment.

Motivation

With the **growing levels of infrastructure development**, sensitive work environment including height and heavy machinery .

This trend introduces serious **health and life challenges**, especially regarding head coverage and skull safety.

Most current systems rely on **Manual checks**, making workers at following risk:

- **Traditional supervision methods** (e.g., on paper)
- **Lack of safety protocols**
- **Raw material crashes** that risk human body exposure

Motivation

- Common methods like **CNN,RCNN**, although easy, are:
  - ➢ **Resource-intensive**, requiring significant CPU cycles
  - ➢ **Suboptimal for real time monitoring** with limited computing power
- Our motivation:
  - ➢ Design a **lightweight, real time, Helmet monitoring framework**
  - ➢ Use **efficient YOLO V5(M)**, a mini yet efficient version   ideal for real time monitoring
  - ➢ Ensure **lesser computation**, **by using 180 degree FISHEYE Lens**

Literature Survey

Author Name	Paper Title	METHODOLOGY	PROS	CONS	Conclusion
X. Ma, K. Ji, B. Xiong, L. Zhang, S. Feng, and G. Kuang **Link:** https://ieeexplore.ieee.org/document/9573455	LightYOLOv4: An edge-device oriented target detection method for remote sensing images	Light-YOLOv4 is obtained from YOLOv4 through sparsity training, channel and layer pruning, knowledge distillation, and quantization. Deployed on NVIDIA Jetson TX2, experiments on SSDD and Gaofen Airplane datasets validate its efficacy for edge devices.	Light-YOLOv4 significantly reduces model size, parameter size, and FLOPs, increasing detection speed by 4.2x with minimal accuracy loss. It outperforms SSD and FPN in suitability for edge devices.	While Light-YOLOv4 offers enhanced efficiency and speed, there's a slight reduction in detection accuracy, though minimal.	Light-YOLOv4 presents a promising solution for edge device deployment, offering substantial reductions in model complexity and computational requirements without compromising detection performance significantly.

F. Wu, G. Jin, M. Gao, Z. HE and Y. Yang) **Link:** **https://ieeexplore.ieee.org/document/8743246**	Helmet Detection Based On Improved YOLO V3 Deep Model	This paper improves helmet detection by enhancing YOLOv3 with a DenseNet backbone for feature extraction, forming the YOLO-Densebackbone CNN. The model performs well on stained, partially occluded, and low-resolution helmets.	YOLO-Densebackbone improves detection accuracy by 2.44% compared to traditional YOLO V3, maintaining the same detection rate. It enhances helmet detection, ensuring safe construction.	Drawbacks include increased computational complexity or training time due to the replacement of the backbone network.	Ihe development of YOLO-Densebackbone offers practical significance by improving helmet detection accuracy, particularly in challenging scenarios such as stained or partially occluded helmets. This advancement contributes to enhanced safety measures in construction environments.
X. Long, W. Cui and Z. Zheng **Link:** **https://ieeexplore.ieee.org/document/8729039**	Safety Helmet Wearing Detection Based On Deep Learning	This paper employs a deep learning approach using a single shot multi-box detector (SSD) for accurate safety helmet wearing detection. To address SSD's limitations in detecting small objects like safety helmets, a novel system is proposed and implemented.	The deep learning approach with SSD enhances accuracy in safety helmet detection, particularly in scenarios like power substations where safety is critical. The proposed system demonstrates efficiency and effectiveness in detecting small objects.	limitations include the need for extensive training data and computational resources for training deep convolutional neural networks like SSD.	The research presents a practical solution for safety helmet wearing detection using deep learning with SSD. Extensive experimental results in power substations validate the system's efficiency and effectiveness, highlighting its importance in ensuring worker safety.

Problem Statement

- Existing safety helmet supervision in construction sites relies on manual checks, leading to inefficiencies and compromising worker safety.
- Traditional supervision methods fail to promptly detect instances of safety helmet non-compliance, increasing the risk of accidents and injuries.
- The vast and complex nature of construction sites makes it challenging to ensure consistent adherence to safety protocols, particularly regarding safety helmet usage.
- There is a pressing need for advanced monitoring systems, such as deep learning-based computer vision, to enhance safety protocols and protect the well-being of construction workers.

Objective

Project Objectives

1. **Lightweight Detection Model**
   - Develop YOLO-M by enhancing YOLOv5s with MobileNetv3 backbone.
   - Reduce parameters & size for deployment on edge devices.
2. **Improve Accuracy for Small & Occluded Targets**
   - Introduce **Res-FPN** for better feature fusion of small targets.
   - Add **BiCAM** attention for focusing on dense objects & reducing noise.
3. **Model Evaluation & Validation**
   - Benchmark YOLO-M against YOLOv5s & other detectors.
   - Metrics: mAP, Recall, Parameters, Inference Speed

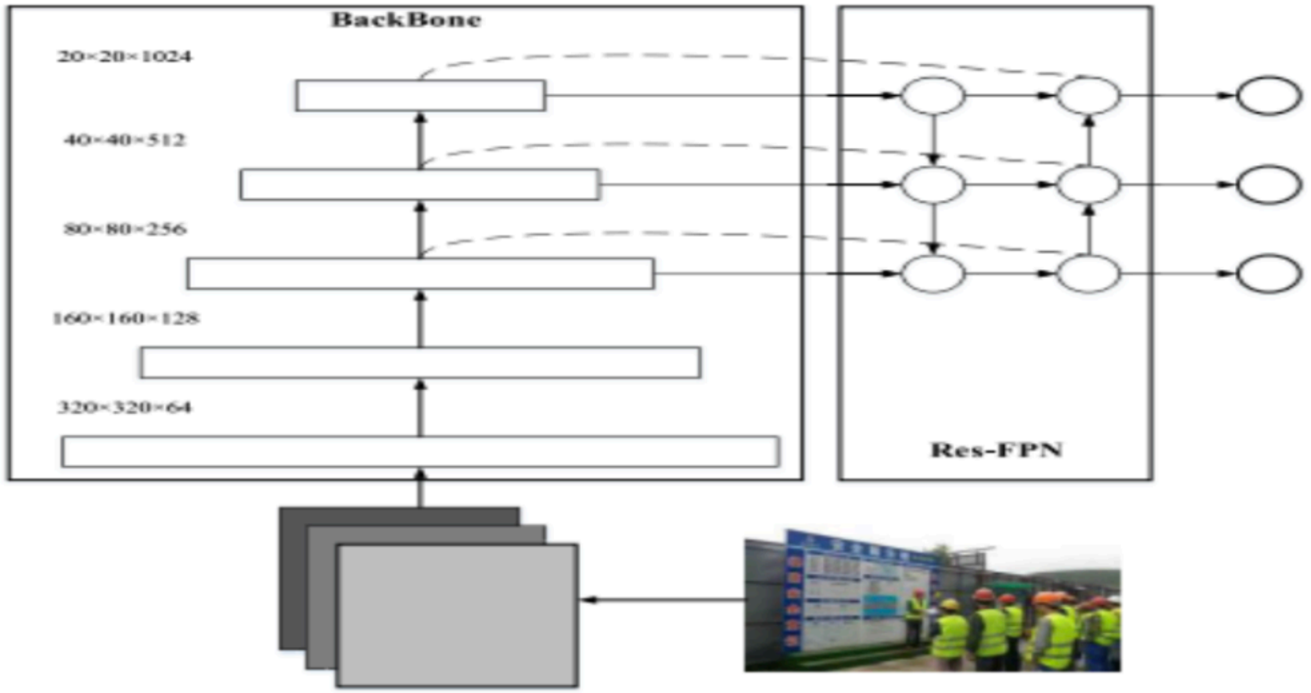


Fig: **System Architecture**

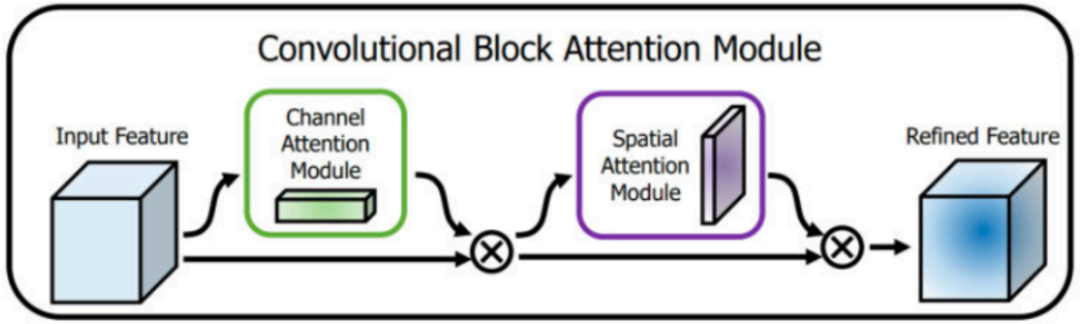
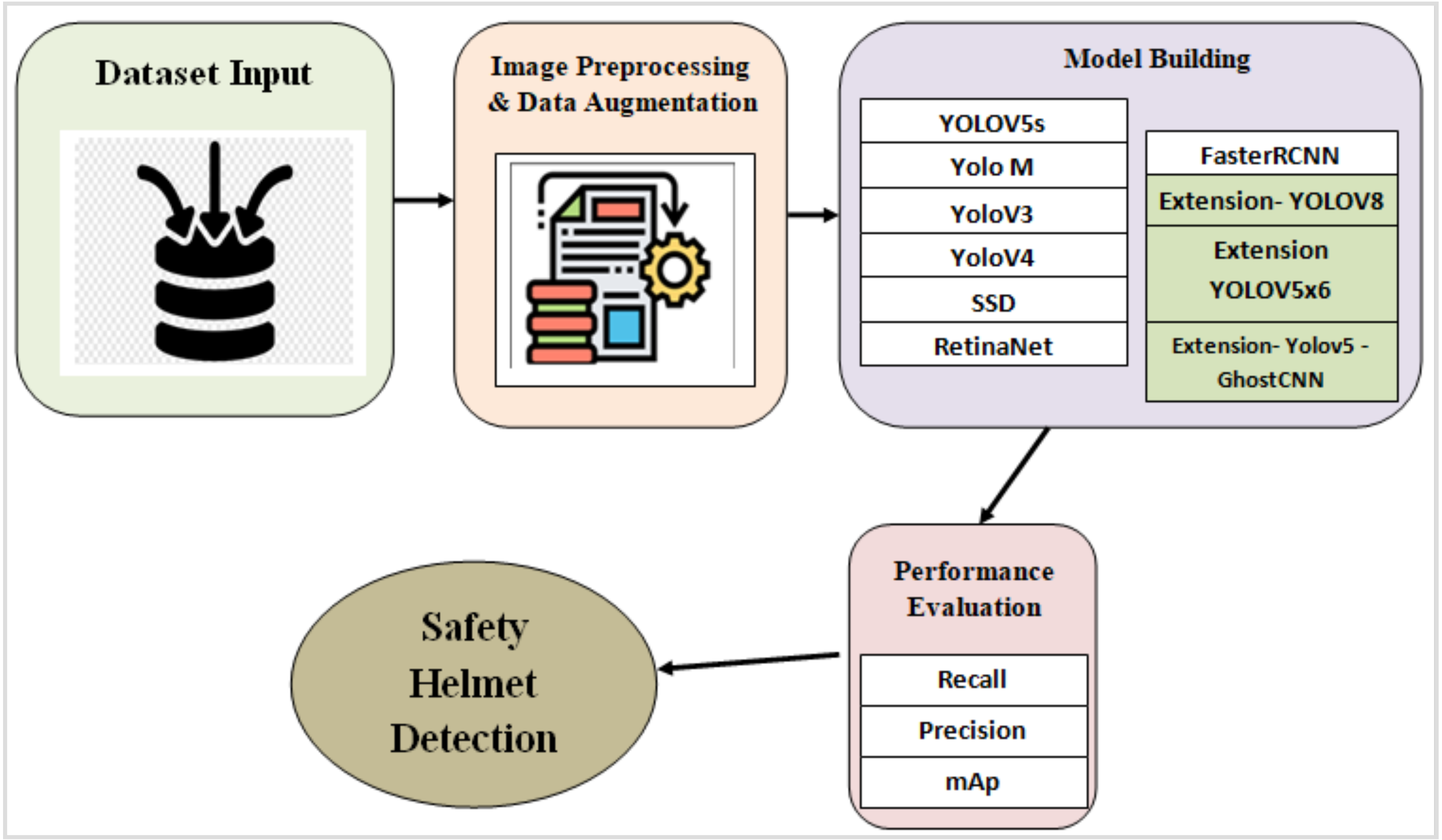


Fig : **CBAM Module**

Methodology



Simulation Platform and Requirements

Software Requirements

1.	Software	:	Anaconda
2.	Primary Language	:	Python
3.	Frontend Framework	:	Flask
4.	Back-end Framework	:	Jupyter Notebook
5.	Database	:	Sqlite3
6.	Front-End Technologies	:	HTML, CSS, JavaScript and Bootstrap4

Hardware Requirements

1.	Operating System	:	Windows Only
2.	Processor	:	i5 and above
3.	Ram	:	8GB and above
4.	Hard Disk	:	25 GB and above

Conclusion

- The project creates an automated safety helmet detection system, actively monitoring construction sites in real time to decrease the risk of head injuries.
- Employing advanced algorithms like YOLO variants, YOLO-M, RetinaNet, and FasterRCNN, the project thoroughly explores safety helmet detection methodologies.
- **The extension into YOLOv5x6 and YOLOv8 demonstrates a commitment to robust safety predictions. Flask integration with user authentication streamlines testing and validation.**
- From refining algorithms to practical Flask applications, the project significantly contributes to construction. Automating safety monitoring processes supports managers and workers, fostering a safer work environment.

References

- X. Ma, K. Ji, B. Xiong, L. Zhang, S. Feng, and G. Kuang, “LightYOLOv4: An edge-device oriented target detection method for remote sensing images,” IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens., vol. 14, pp. 10808–10820, 2021, doi: 10.1109/ JSTARS.2021.3120009
- F. Wu, G. Jin, M. Gao, Z. HE and Y. Yang, "Helmet Detection Based On Improved YOLO V3 Deep Model," 2019 IEEE 16th International Conference on Networking, Sensing and Control (ICNSC), Banff, AB, Canada, 2019, pp. 363-368, doi: 10.1109/ ICNSC.2019.8743246.
- X. Long, W. Cui and Z. Zheng, "Safety Helmet Wearing Detection Based On Deep Learning," 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chengdu, China, 2019, pp. 2495-2499, doi: 10.1109/ITNEC.2019.8729039.
- S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, 1 June 2017, doi: 10.1109/ TPAMI.2016.2577031.
- R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2014, pp. 580–58.

Thank you