INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD)
© 2026 IJNRD | Volume 11, Issue 11, November 2026 | ISSN: 2456-4184 | IJNRD.ORG

"VisionShield

AI-Based Foundry Monitoring System"

Snehal Sunil Wagavekar , Sanika Vijay Gurav , Shweta Rajendra Vhannure , Sufiya Balaso Bargir , Samiksha Vinod Gaikwad
Computer Science & Engineering ,
D. Y. Patil College of Engineering & Technology, Kasaba Bavda , Kolhapur, India .

Abstract : Vision-Shield provides an AI-based monitoring solution for foundries, intended to make real-time detection of unsafe events and process violations easier trough a single web dashboard. The system uses a YOLO11 model to detect heat overflow events and track process activities in video , while OpenCV marks each detection with bounding box, label , and confidence score. A rule engine compares process readings such as temperature, heating duration, and material weight against predefined limits and flags any violation. The backend is built in python , FastAPI connects the trained models to the interface, and React.js dashboard displays violations and process timing. Earlier manual monitoring was slow, missed events, and left no reliable record. The proposed system detects violations automatically and saves each one with its timestamp and evidence frame, making it more suitable for real-time foundry monitoring.

Keywords : Image Extraction, YOLO11, Computer Vision, Rule engine

INTRODUCTION

We introduce VisionShiled , an AI-based monitoring system our team built to watch foundry operations and detect abnormal conditions automatically. The need for such a system comes from the nature of foundry work. Molten metal, very high temperatures, and strict limits on heating time and material weight leave little room for error. In most plants, these steps are still checked by people, so a heat overflow or a process violation can be missed or noticed too late, and there is often no reliable proof of what happened. VisionShield was designed to handle this by analyzing video and process readings continuously and flagging problems as they occur.

The architecture of VisionShiled is organized into four layers. The data input layer takes in CCTV footage from the foundry and process sensor readings. At present, the sensor values are simulated , and the design allows real ESP32-based sensors to replace them later. The application layer is written in Python and is built around a FastAPI backend that provides the upload, analysis, and result endpoints .Inside this layer, OpenCV processes the video frame by frame, and two trained YOLO11 models detect heat overflow events and track process activities. A rule engine then compares the detections and sensor readings against predefined thresholds. Whenever a rule is broken , an evidence generator saves the annotated frame with its timestamp, and the results store keeps every violation for review.

From the start , we had to think about the end users , who are plant supervisor, operator and admin , not programmers. They need something simple that works in a browser. For this reason, the frontend is a React.js dashboard that communicates with the backend through a REST API over HTTP. It has three main parts : Live monitoring through uploaded videos ,an evidence viewer that shows annotated frames with bounding boxes , labels, and summary panel that shows violation counts, charts, and process timing. The user does not need to understand the models or the rules behind them, because everything is controlled through this one interface.

Our main objective was to build a tool that is accurate yet easy to use. The system was engineered to be modular, so the detection models, the rules, and the sensor inputs can each be improved without changing the rest. The result is a platform that replaces slow manual checking with automatic detection and keeps a clear dashboard, VisionShield can make foundry monitoring more reliable and transparent , and the same approach can be extended to other heavy industries where safety and process limits matter.

Literature Review

A. Industry 4.0 and Foundry Process Monitoring

Industry 4.0 brings sensors, edge computing, and data analysis into factories so that decisions can be made faster. In casting, Denker et al. proposed an edge-based monitoring and control system built on the BaSyx framework for real-time process monitoring[1]. This shows that automatic monitoring can speed up decisions in a foundry. However, the work stays focused on temperature and the casting setup. It does not cover material tracking or weight monitoring[1].

B. Vision-Based Process Monitoring

Computer vision lets a system watch a process continuously without a person standing next to it. Ravi et al. built a deep learning model that reads video from a camera inside an aluminium melting furnace, detects whether scarp is present, and notifies operators when melting ends, with accuracy of up to 99% [2]. Yan and Wang combined YOLOv3 with VGG16 to monitor operator actions in a manufacturing workshop, reaching more than 96% recognition accuracy with an average time error under one second[3]. Both studies show that video can follow a process reliably. Their limits are that
[2] covers only aluminium scrap and melting completion and does not measure weight or temperature, while[3] works only for predefined worker actions and does not use IoT sensor data or furnace temperature.

C. Machine Learning for Defect Prediction

Many researchers use process data to predict casting quality. Uyan et al. used a cloud-based Industry 4.0 data system with an XGBoost model to predict porosity defects in low-pressure die casting[4]. Pachandrin et al. compared Random Forest, SVM, and AutoML models for predicting defects in magnesium high-pressure die casting[5]. Vijaykumar and Reddy reviewed machine learning methods for foundry defect prediction, including neural networks, decision trees, and ensemble methods[6]. These works show that process parameters can warn about quality problems. However,[4] focuses on aluminium die casting,[5] on magnesium die casting, and[6] is review that does not propose a monitoring system.
None of them offers real-time monitoring, weight measurement, or stored evidence.

D. Supporting Software

YOLO detects object in an image in a single pass, which makes real-time video analysis possible[7]. We use YOLO11, the latest Ultralytics version, for detection [8]. OpenCV reads the video, processes each frame, and draws the bounding boxes, labels, and confidence scores on the detections[9]. FastAPI connects the trained models to the interface, React.js builds the dashboard, and a rule engine compares readings such as temperature ,heating duration, and material weight against predefined limits.

E. Supporting Hardware

The system needs CCTV or IP cameras for foundry video and sensor for temperature, heating time , and weight. In the current version the sensor values are simulated, and the design allows ESP32 microcontrollers to send real readings later. A computer runs the detection models and the backend.

PURPOSED METHODOLOGY

The proposed system is organized as four layers: a client layer, a frontend layer ,an application layer, as shown in Fig. Each layer has one clear responsibility and tasks only to its immediate neighbours. This keeps the design easy to maintain.

A. System Architecture :

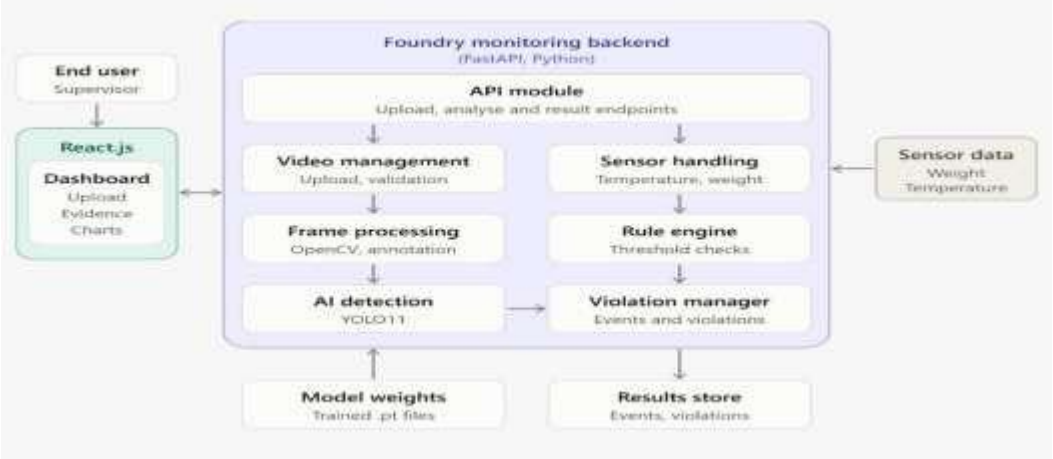


Fig 1. System Architecture of VisionShield

A. Client Layer: The client layer is where people meet the system. Authorized personnel open a web dashboard in an ordinary browser, so nothing has to be installed on their machines. The system defines three roles, administrator, supervisor and operator and what each person can see or change depends on the role.

B. Frontend Layer: The dashboard is built with React.js and has three main parts.A video upload module lets the user pics a foundry recording for analysis. An evidence viewer shows the annotated frames , where detected events are marked with labelled bounding boxes. A summary panel gives the overall picture through violation counts and charts. The frontend sends request to the backend over a REST API using HTTP and displays whatever comes back.

C. Application Layer: Most of the work happens in python. A FastAPI backend receives the uploaded video and exposes the endpoints for starting an analysis and fetching results. The video is first passed to OpenCV, which reads it and splits it into individual frames. Those frames go to the detection stage, which uses YOLO11. We use two separately trained models, one for general foundry objects and one dedicated to heat overflow, so that each can be tested and improved on its own.

D. Data input layer: This layer supplies the raw material for the whole pipeline. The first source is recorded clips uploaded by the user. The second source is sensor data covering temperature and weight. At this stage the readings are simulated. The design allows for a later move to ESP32-based sensors.

E. Data Flow: Analysis starts when a user uploads a clip through dashboard. The backend passes it to OpenCV for frame extraction,YOLO11 examines the frames ,and the rule engine checks the results against the sensor values.Any violation are turned into stored evidence and returned through the RestAPI. The dashboard then presents the annotated frames, counts and charts to the user.

B. Communication Workflow :

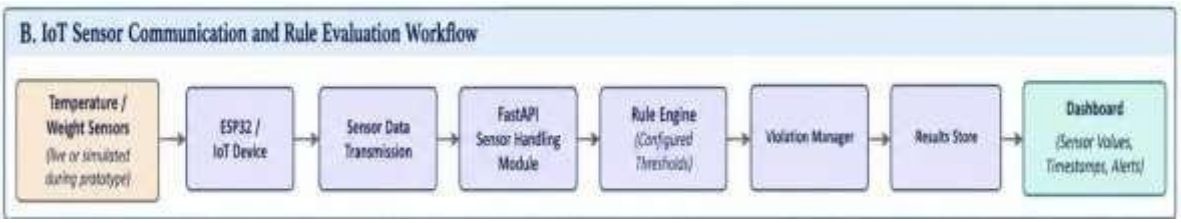


Fig2. IoT Sensor Communication & Rule Evaluation Workflow

1. Sensor Monitoring and Alert :

Temperature and weight readings reach the FastAPI backend, and each one is saved with its time. The rule engine compares the values with the allowed ranges and marks the reading Normal or Abnormal. An abnormal reading creates an alert of type High Temperature, Low Temperature, High Weight or Low Weight. The alert

shows the current value, the expected range, and the time of detection. The sensor analytics page lists every reading and draws the temperature and weight trends with the minimum and maximum limits marked on the cart.

2. AI PPE Detection :

On the AI Safety Monitoring page, the user uploads an image or video. The backend runs the trained YOLO11 model, which detects person, helmet, vest, gloves, and boots .The result is returned as an annotated image with a labelled box and confidence score for every detections. A summary table shows, for each class, the number of detections, the average confidence, and the confidence range. The user can filter the results by class name and by a minimum confidence, and can download the annotated result.

3. Process Detection :

On the Process Detection page, the user uploads a foundry video. The backend creates a background job, so the user can move to other pages while the video is processed. Each job shows its status(Processing or Done) and its progress, with a Track button for running jobs and Load Result button for finished jobs. OpenCV extracts the frames, and the trained YOLO model detects the process steps. The detected order of steps is checked against the required sequence. Magnesium addition is mandatory, and if it is missing the system raise a Critical alert, saves it in MongoDB, and sends an email to the configured process alert address.

4. Live Monitoring :

The Live Monitoring page shows all CCTV feeds together in a multi-camera grid, next to the current temperature and weight with their status. Health indicators show whether the sensor, the camera, the API, and the database are working, and the page refreshes automatically every second.

C. Dataset and Model Training :

The system uses two separately trained YOLO models. The PPE model is based on YOLO11 and detects five classes

: person, vest, helmet, vest, gloves, and boots. The process model detects the steps of foundry process in video. The training data was collected from Roboflow, labelled using Roboflow and split into training, validation, and test sets in the ration[70:20:10]. The trained images are saved as .pt files, which the backend loads when it starts.

Table 1. Details of the Trained Models

Item	PPE Model	Process Model
Architecture	YOLO11	YOLO11
Classes	Person, helmet, vest, gloves, boots	Person, helmet, vest, gloves , boots
Total images	1500	100
Train/ Val/ Test	825	50
Epochs		
Image size	600*400	554*1300

D. Sensor Handling & Rule Engine :

The rule engine is the decision-making part of the system. It compares the sensor readings and the detected process steps with predefined conditions, shown in Table 2. A reading with both values out of range creates one alert with two tags.

Table 2.Rules Checked by the Rule Engine

Parameter	Source	Allowed Limit	Alert Raised
Temperature	Sensor	1500-1600°C	High or Low Temperature
Material Weight	Sensor	500-650 kg	High or Low Weight
Magnesium Addition	Process detection (YOLO)	Mandatory	Critical alert if skipped

E. Alert Management and Notifications

The Alert page lists every alert with its type, the current value, the expected range, the time of detection, and its status, Active or Resolved. Four counters at the top show the active alerts, the alerts resolved in the last 24 hours, the temperature issues, and the weight issues. The user can filter alerts by status and date, and can resolve an active alert with one click. A resolved alert keeps the time of resolution and the name of the user who resolve it, so the record stays complete for later review.

The setting page controls the notifications. The administrator can enter an email address and a mobile number, switch email and SMS alert on or off, and send text message. A separate process alert email, which only the administrator can change, receives the alert when the magnesium step is skipped.

F. Role-Based Access and User Management :

The system has three roles. The administrator has full access, including the Settings, Maintenance Management, and User Management modules. The supervisor and the operator have standard access. The operator menu shows only the Dashboard, Alerts, AI Safety Monitoring, Sensor Analytics, Live Monitoring, and Process Detection modules. In user, see each account's role and status, and manage the accounts. The main administrator account is protected and cannot be changed.

G. Algorithms :

1. Sensor Monitoring & Alert Generation

1. Input: sensor reading(temperature T, weight W),limits(T_min, T_max, W_min, W_max).
2. Save the reading with its time.
3. If T<T_min or T>T_max, tag a temperature issue(Low or High).If W<W_min or W>W_max, tag a weight issue(Low or High).
4. If no tag exists, mark the reading Normal. Otherwise mark it Abnormal and create an alert with tags, the current values, and the expected ranges.
5. Show the alert on the dashboard and send the email or SMS if enabled.
6. When a user resolves the alert, save the resolution time and user name.

2. Process Sequence Detection

1. Input : Video V, trained process model W, required step list S.
2. Create a background job for V and set its status to Processing.
3. Extract the frames of V with OpenCV.
4. For each frame, run the model and record the detected step with its confidence.
5. After the last frame, compare the detected step with S.
6. If the magnesium step is missing, save a Critical alert in MongoDB and send the process alert email.
7. Set the job status to Done and save the result so that it can be loaded later.

H. Implementation Environment :

Table 3. Software and Hardware Used

Item	Details
Programming Language	Python 3.13.5
PPE detection	YOLO11
Process detection	YOLO11
Video processing	OpenCV 5.0.0
Backend	FastAPI
Frontend	React.js
Database	MongoDB
Sensor data	ESP-32
Processor	i5
RAM/OS	16GB/Window

I. Performance Measures :

Detection quality is measured with precision, recall, and mAP@0.5 :

Precision = TP/(TP+FP) Recall = TP/(TP+FN)

Where TP is the number of correct detections, FP is the number of wrong detections, and FN is the number of missed objects.

The processing time for one video frame is:

T_frame = T_read+T_detect+T_rule+T_store and the processing speed is FPS=1/T_frame.

The sensor alert time is the delay between a sensor reading and its alert on dashboard: T_alert = T_transmit + T_rule + T_display

IV. RESULTS AND DISCUSSION :

A. User Management & Access Module :

The user management module controls who can use the system. In our test setup there were five registered users : one administrator(admin),two supervisors(supervisor1 and supervisor),and two operators(operator1 and operator2), and all accounts were active. The administrator account is marked Protected and has full access, while the other accounts have standard access and can be managed by the administrator. The menu also depends on the role. The administrator sees all the nine modules, while the operator sees only six :Dashboard, Alerts, AI Safety Monitoring, Sensor Analytics, Live Monitoring, Sensor Analytics, and Process Detection. Settings, Maintenance Management, and User Management are not shown to the operator.

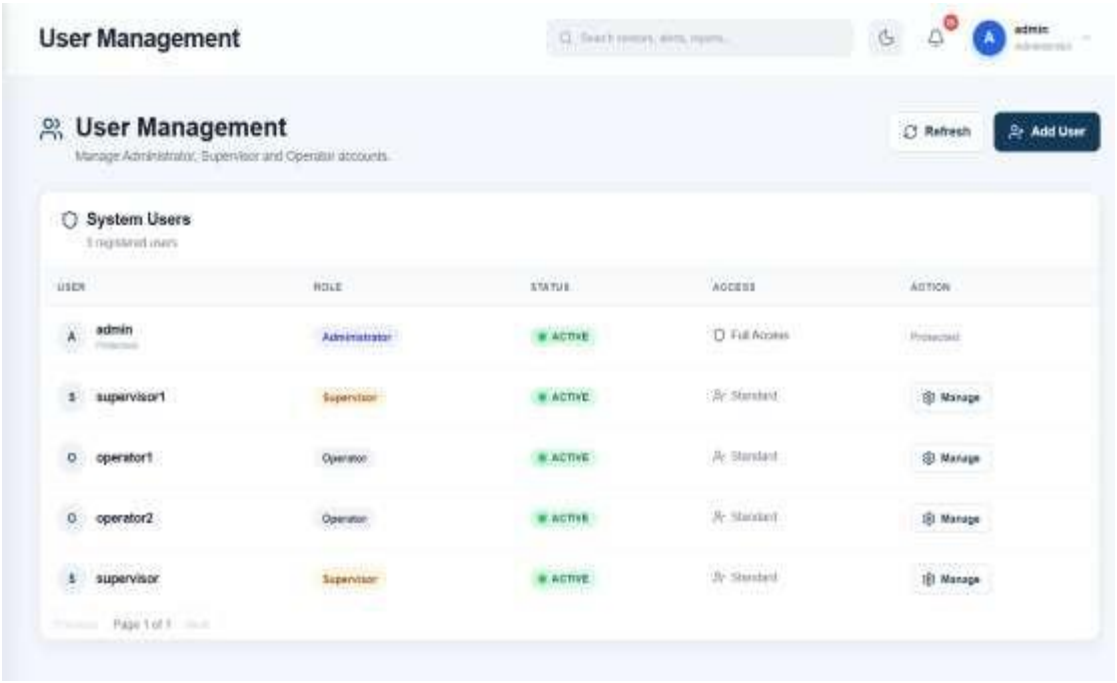


Fig.3(1). Admin Interface

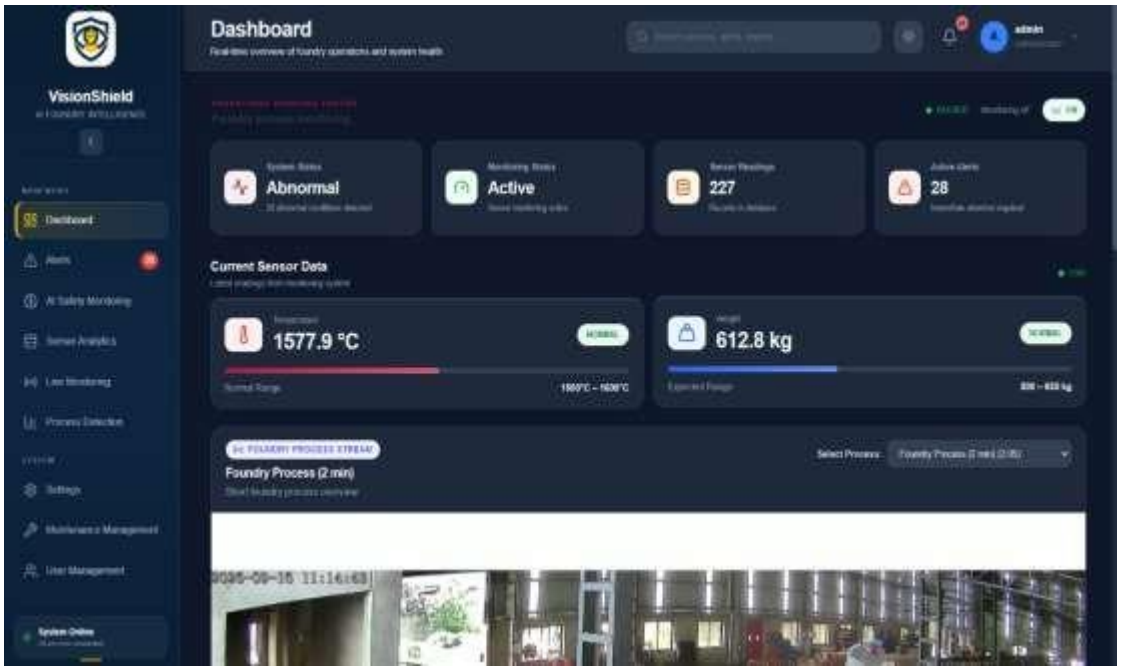


Fig. 3(2). Supervisor Interface

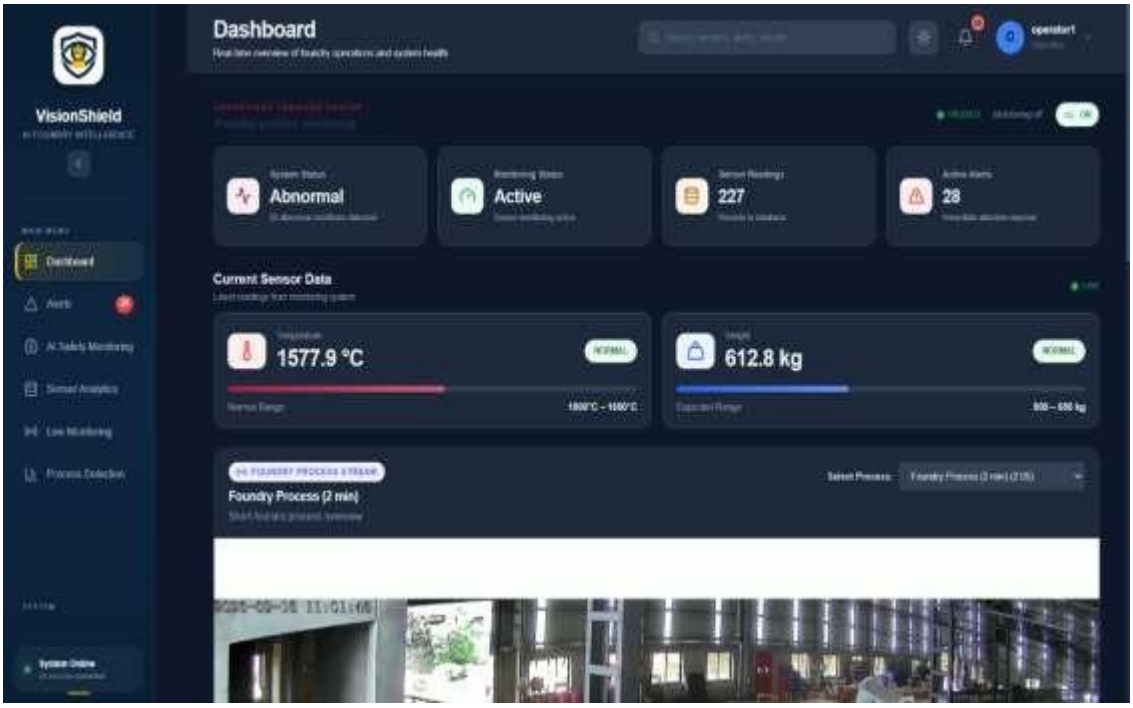


Fig. 3(3). Operator Interface

B. Dashboard and Live Monitoring :

The Dashboard provides a centralized overview of foundry operations, displaying system status, monitoring status, sensor readings, and active alerts. The Live Monitoring module displays multiple CCTV camera feeds along with real-time temperature, material weight, and system health indicators.

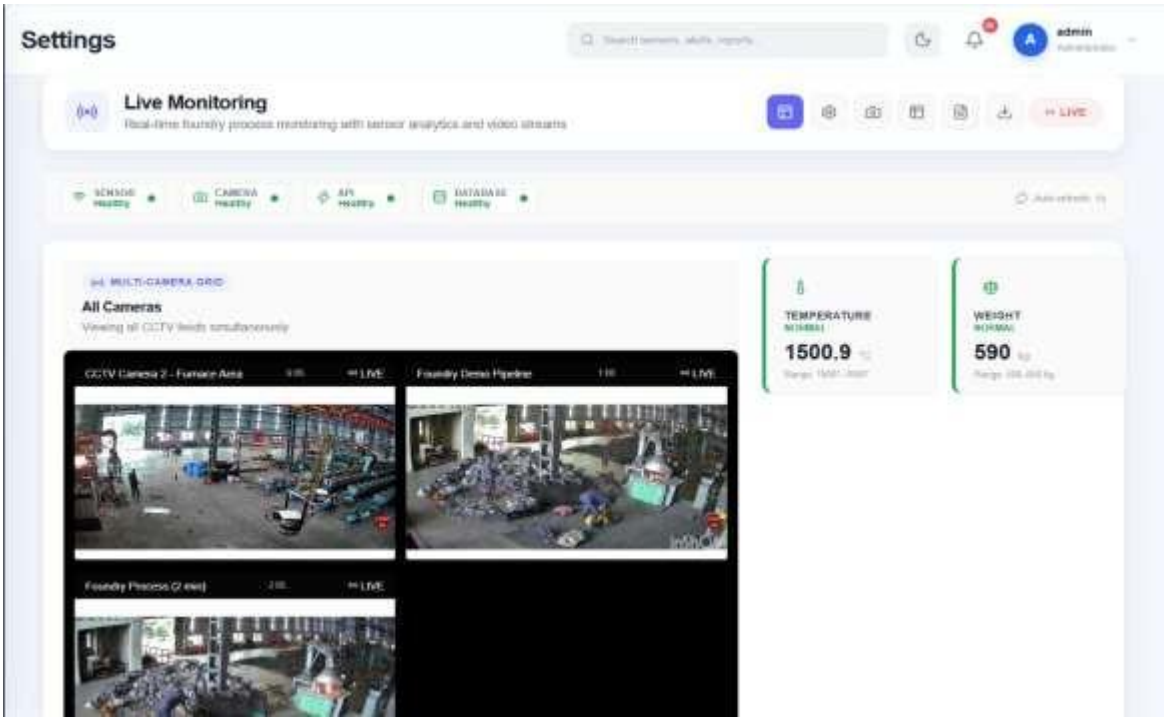


Fig. 4. Live Monitoring Dashboard

C. Sensor Analytics

The Sensor Analytics module continuously monitors temperature and material weight readings an and compares them with predefined operating limits. It displays historical sensor data in tabular a and graphical formats, helping users identify abnormal conditions such as excessive temperature or material weight.

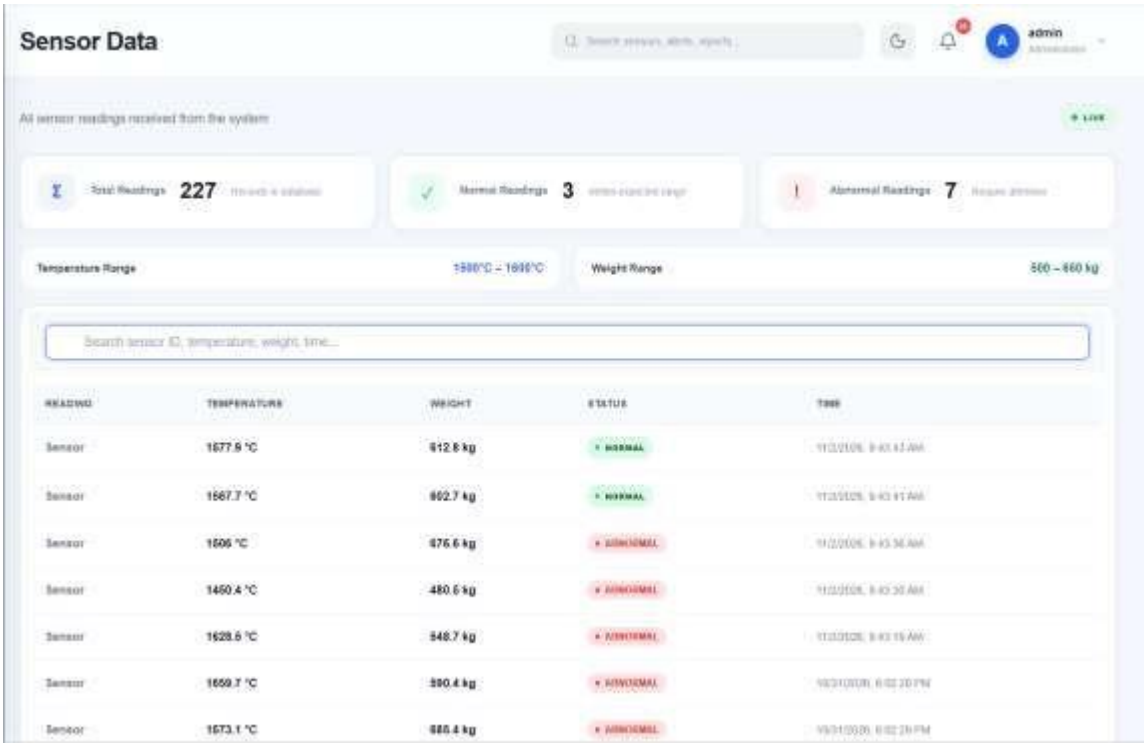


Fig. 5(1). Sensor Data



Fig. 5(2). Sensor Monitoring Graph

D. Alert Management :

The Alert Management module automatically generates alerts whenever temperature or material weight exceeds the specified operating limits. Users can view active alerts, check abnormal readings, and resolve issues, while system maintains a history containing detection time, resolution time, & the responsible user.

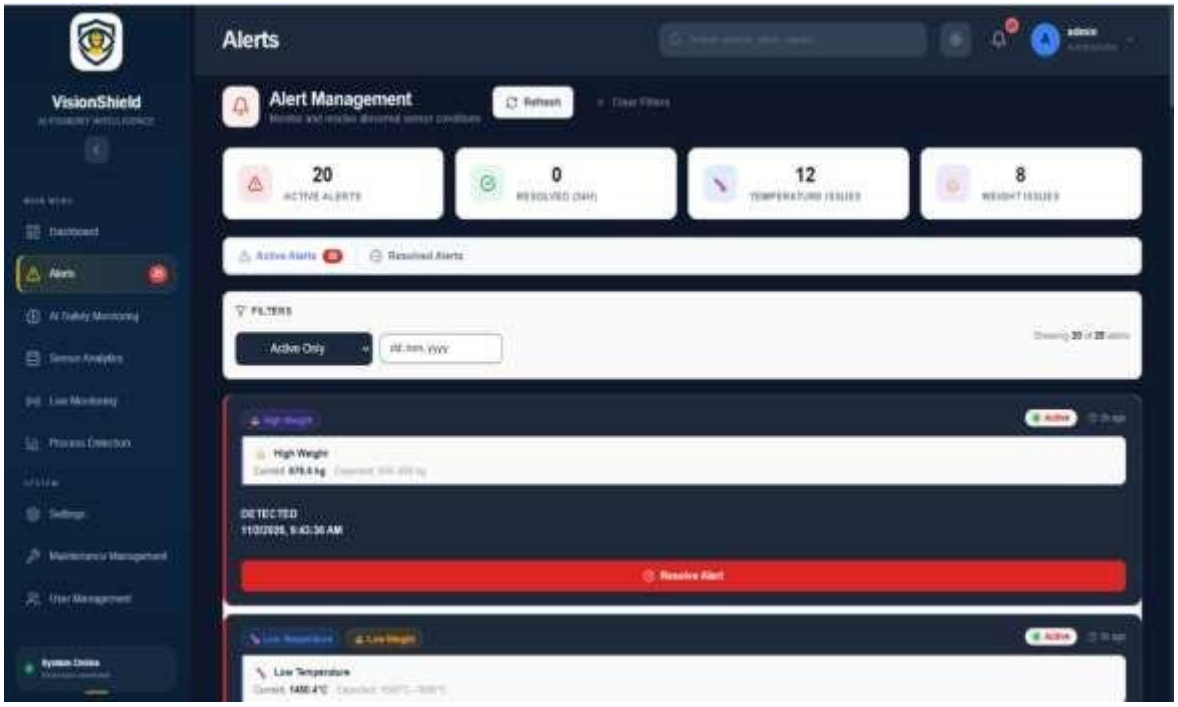


Fig.6(1).Alert before Resolve

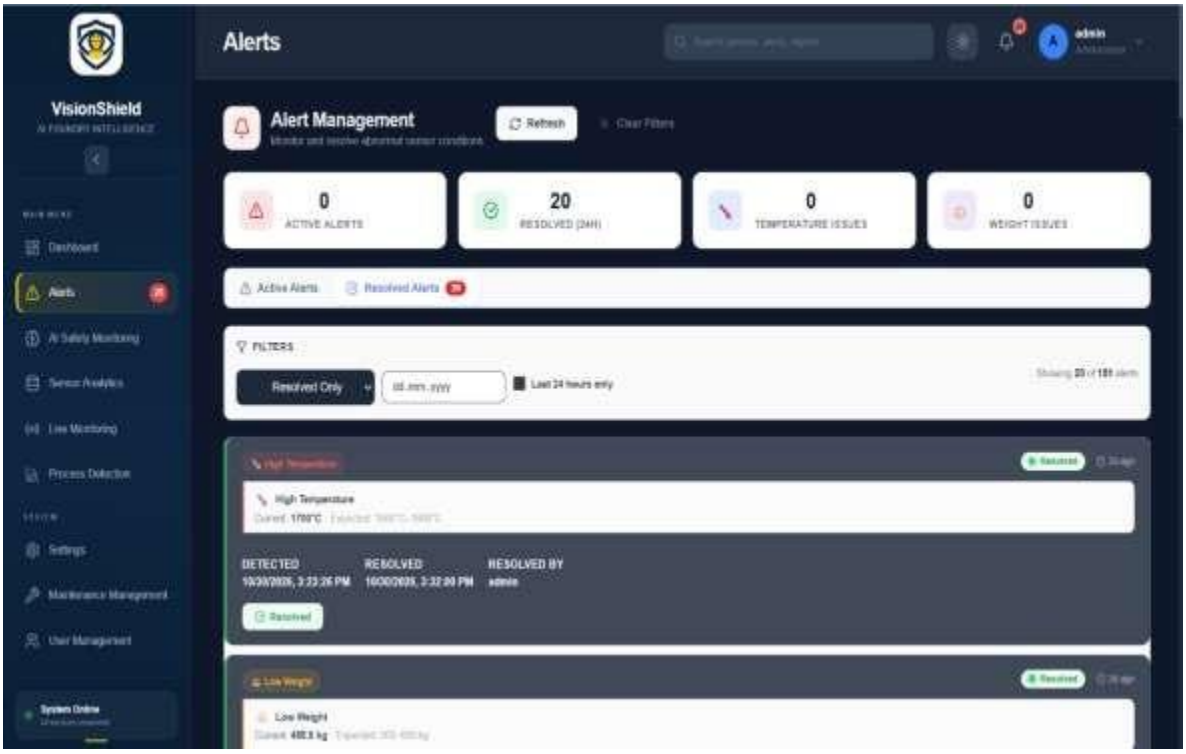


Fig.6(2).Alert after Resolved

E. Alert Notifications :

The Alert Notification module provides automatic communication whenever abnormal conditions are detected in foundry. Administrators can configure email and SMS notifications, manage contact details, and receive separate critical email notifications when the mandatory magnesium addition process is skipped.

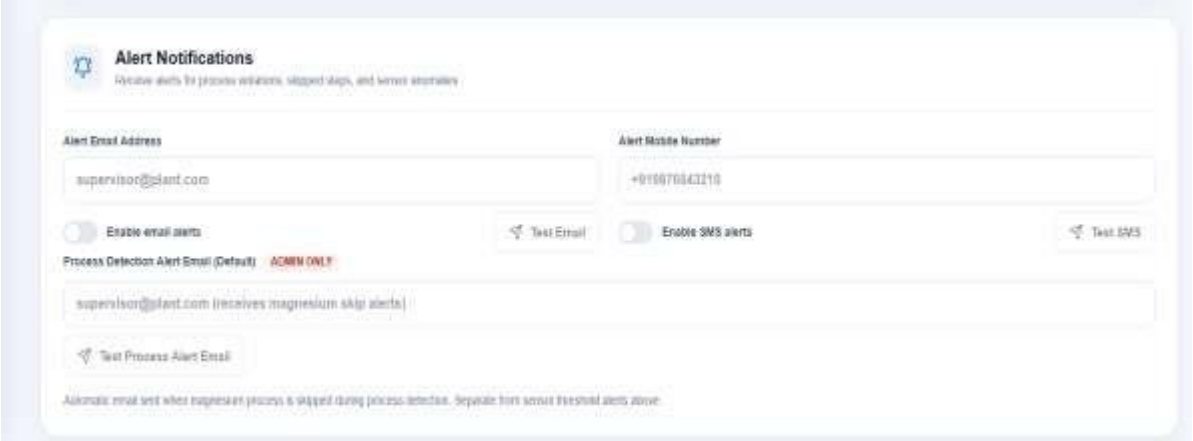


Fig.7.Alert Notifications

F. AI PPE Detection :

The AI PPE Detection module uses the YOLO11 object detection model to identify workers and their personal protective equipment, including helmets, safety vests, gloves, and boots. It displays detected objects using bounding boxes and confidence scores, helping supervisors identify potential safety compliance issues.

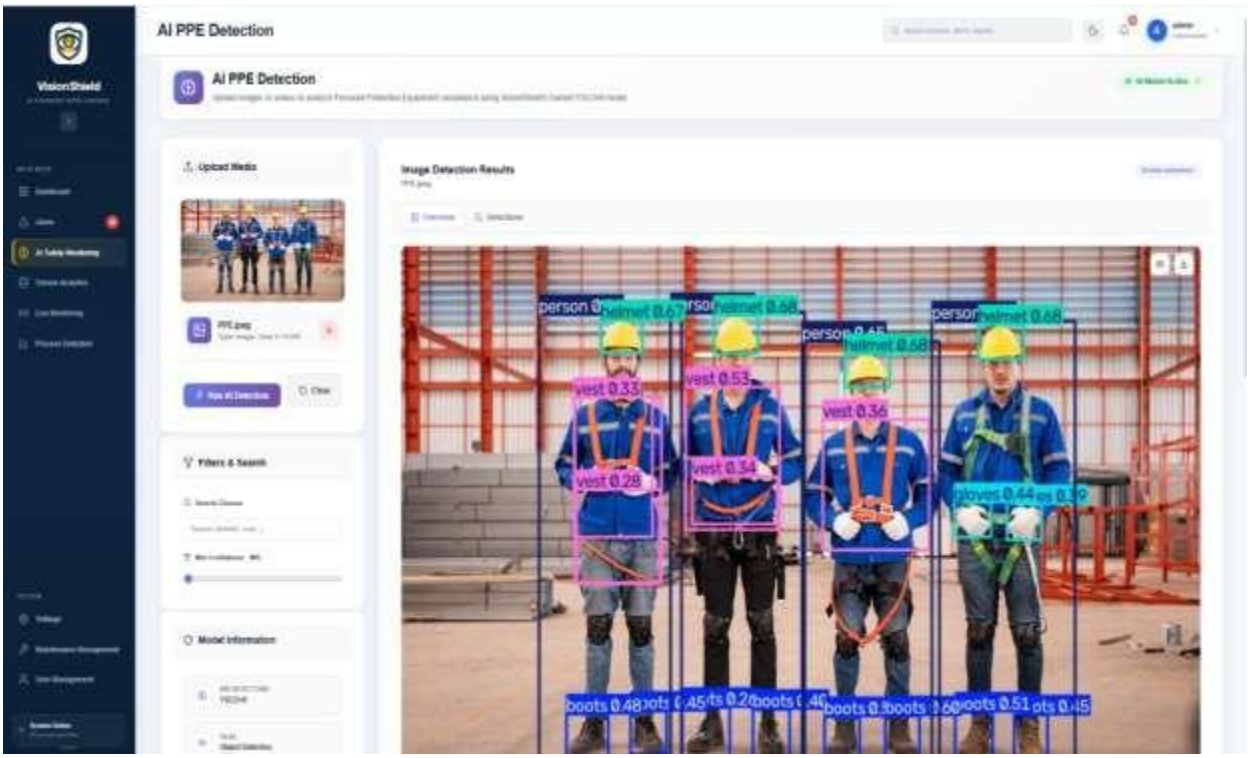


Fig.8.PPE Detection

G. Process Detection :

The Process Detection module uses a trained YOLO model and OpenCV to analyze uploaded foundry videos and identify different stages of the manufacturing process. It verifies the required sequence of operations and generate a Critical alert if the mandatory magnesium addition step is missing .

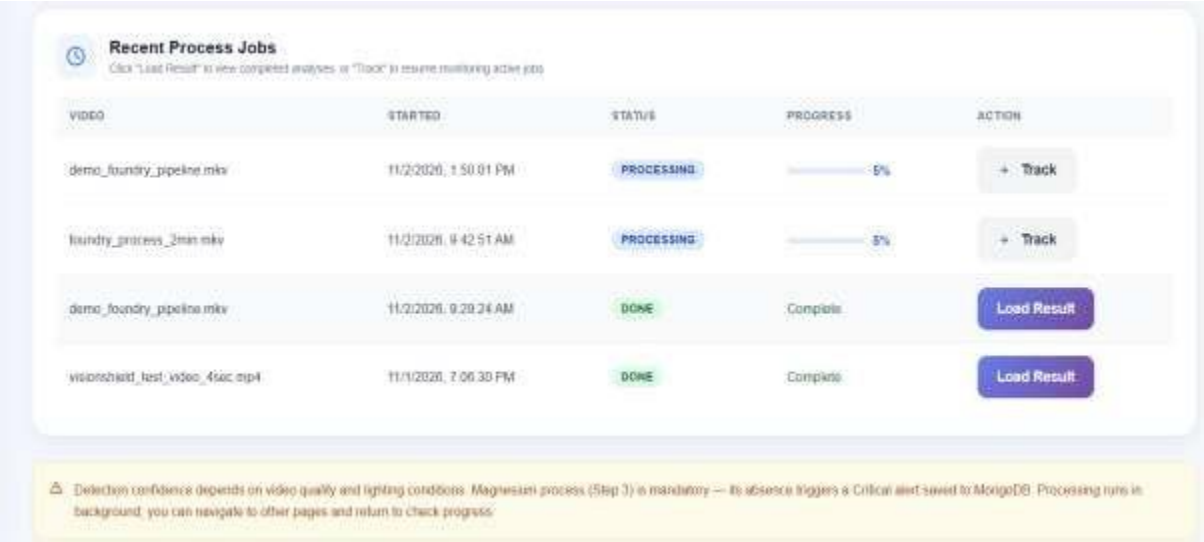


Fig.9.Process Detection

H. Detection Performance and Processing Time :

This module evaluates the performance of trained PPE and process detection models using precision, recall, and mean average precision. Processing time and frames per second are also measured to understand the system's detection efficiency and suitability for real-time monitoring.

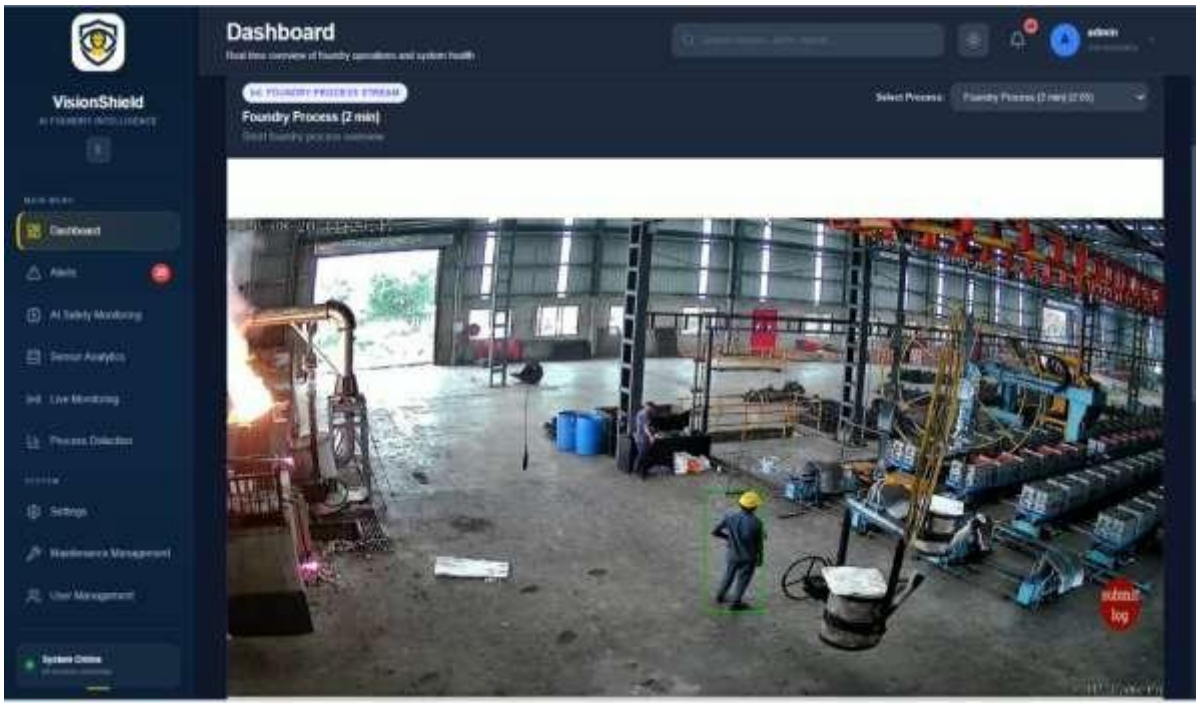


Fig.10.Fpundry Process

I. Comparative Analysis of VisionShield and Related Works Table 4. Comparative Analysis

Feature	**Sensor-based monitoring[1]**	**Vision-based monitoring [2],[3]**	**ML detect prediction[4]-[6]**	**Proposed VisionShield**
Data source	Temperature and process data	Video	Process parameters	Sensor readings, images, and video
Real-time monitoring	Yes	Yes	No	Yes
Temperature and weight limit checks	Temperature only	No	Used as model inputs	Yes, with Normal or Abnormal status
PPE detection	Not reported	Not reported	Not reported	Yes
Process step check	Not reported	Melting completion[2],work-shop actions[3]	No	Yes (magnesium step mandatory)
Alert records and notifications	Not reported	Operator notifications[2]	No	Alerts with resolve history, email, and SMS
Role-based access	Not reported	Not reported	Not reported	Yes(three roles)

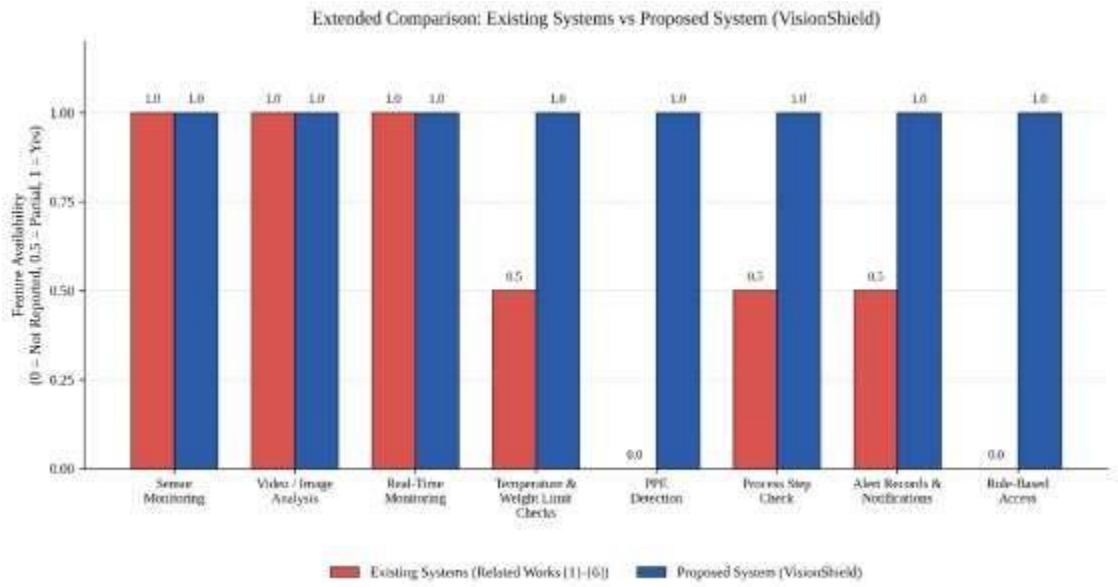


Fig11. Extended Comparison Graph

K. Conclusion :

This study presented VisionShield, an AI-based foundry monitoring system that combines sensor monitoring, PPE detection with YOLO11, process step detection, and React.js dashboard with role-based access. The system checks temperature and weight against fixed limits, raise alerts that can be resolved and sent y email or SMS, and raises a Critical alert when the mandatory magnesium step is skipped. In our tests, it reached a 25 FPS. This makes foundry monitoring faster, more transparent, and less dependent on manual checking.

In future work, we plan to replace the simulated sensor values with real ESP32-based sensors, connect live CCTV streams, and train the models on more foundry data and more process steps. The same approach can also be used in other heavy industries where safety and process limits matter.

REFERENCES

[1] J. Denker, V. Iannino, C. Laudenberg, A. Tenner, M. Jelali, and M. Daun, "Improved temperature monitoring and controlof production lines in casting through BaSyx framework and edge intelligence," in Proc. Int. Joint Conf. Neural Netw.(IJCNN),IEEE World Congr. Comput. Intell., Padua, Italy, 2022, pp. 1–8.

[2] Y. S. Ravi, S. Marelli, P. Chiariotti, M. Colombo, and M. Tarabini, "Scrap monitoring in aluminum melting furnaceusing computer vision and deep learning," in Proc. 2024 IEEE Int. Workshop Metrol. Ind.4.0 IoT (MetroInd4.0&IoT),2024, pp. 280–285, doi: 10.1109/MetroInd4.0IoT61288.2024.10584226.

[3] J. Yan and Z. Wang, "YOLO V3 + VGG16-based automatic operations monitoring and analysis in a manufacturing workshop under Industry 4.0," J. Manuf. Syst., vol. 63, pp. 134–142, 2022.

[4] T. Ç. Uyan, K. Otto, M. S. Silva, P. Vilaça, and E. Armakan, "Industry 4.0 foundry datamanagement and supervised machine learning in low-pressure die casting quality improvement," Int. J. Metalcasting, pp. 414–429, 2022,doi: 10.1007/s40962-022-00783-z.

[5] S. Pachandrin et al., "Data-driven prediction of casting defects in magnesium high-pressure die casting using machine learning," Int. J. Metalcasting, vol. 20, pp. 112–131, 2026, doi: 10.1007/s40962-025-01592-w.

[6] H. K. Vijaykumar and N. R. Reddy, "Review on applications of machine learning for defect prediction and Optimization in foundry processes," Manuf. Technol. Today, vol. 24, no. 1–2, pp. 1–11, 2025, doi: 10.58368/MTT.24.1-2.2025.1-11.

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection,"in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2016, pp. 779–788.

[8] G. Jocher, A. Chaurasia, and J.Qiu, "Ultralytics YOLOv8," 2023. [Online].

Available:https://github.com/ultralytics/ultralytics.

[9] G. Jocher and J. Qiu, "Ultralytics YOLO11," 2024. [Online]. Available: https://github.com/ultralytics/ultralytics

[10] G. Bradski, "The OpenCV library," Dr. Dobb's J. Softw. Tools, 2000.

[11] S. Ramírez, "FastAPI," [Online]. Available: https://fastapi.tiangolo.com. [Accessed: DD Mon.YYYY].

[12] MongoDB, Inc., "MongoDB documentation," [Online]. Available: https://www.mongodb.com/docs. [Accessed: DD Mon. YYYY].

[13] Meta Platforms, Inc., "React," [Online]. Available: https://react.dev. [Accessed: DD Mon. YYYY].

[14] Espressif Systems, "ESP32 series datasheet," [Online]. Available: https://www.espressif.com. [Accessed: DD Mon. YYYY].

Copyright & License:

© Authors retain the copyright of this article. This work is published under the Creative Commons Attribution 4.0 International License (CC BY 4.0), permitting unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.