IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15837

Deep Learning-Based Image Classification and Real-Time Object Detection Using CNN and YOLOv8

Madan S[1], Prem Singh M[2], T Chandraiah[3], Poornima B H[4], Basavanna M[5]

Department of Studies in Computer Applications (MCA), Davangere University, Davangere, Karnataka, India[1]

Department of Computer Science Government College (Autonomous), Mandya, Karnataka, India[2]

Department of Computer science, Yuvaraja College, University of Mysore, Mysore, Karnataka, India[3]

Department of Studies in Computer Science, Davangere University, Davangere, Karnataka, India[4,5]

Abstract: Image classification is one of the fundamental areas of application of Machine Learning and Deep Learning in the field of computer vision. The exponential rise in the number of digital images produced by smartphones, cameras, satellites, and medical devices has made manual classification tedious, time-consuming, and error-prone, while conventional approaches depend on manually engineered features and give inaccurate results for complex images. This work presents a web-based Image Classification and Object Detection System that classifies Natural, Medical, and Satellite images using a trained Convolutional Neural Network and performs real-time object detection using YOLOv8. The system was trained on a multi-domain dataset of 12,783 RGB images spanning 14 classes across three groups, with all images resized to 224 × 224 pixels for the M standard input. It is deployed as a Flask web application in which users register, log in, upload an image, and receive the predicted category together with a confidence score, while every prediction is stored in an SQLite database and made available through a prediction history, categories, performance, and profile dashboard. Experimental results show confidence levels between 85.1% and 100% across the tested classes, confirming that the system performs accurate and efficient image classification with a user-friendly interface and no requirement for manual feature extraction.

Keywords: Image Classification, Deep Learning, Convolutional Neural Network, YOLOv8, Object Detection, Transfer Learning.

I. INTRODUCTION

Image classification is one of the fundamental areas of application of Machine Learning and Deep Learning in the field of computer vision. It involves the process of assigning an input image to one of the available categories based on the features contained within the image. The need for image classification has increased tremendously with the increase in the number of digital images generated from smartphones, cameras, satellites, medical devices, and social media networks.

The traditional approaches to image classification were based on manually engineered features. This process was very tedious and resulted in low accuracy, especially for complex images. Recent developments in the field of Deep Learning have, however, led to automatic methods of image classification using deep neural networks, in which a Convolutional Neural Network learns features directly from the image data and removes the need for extensive feature engineering.

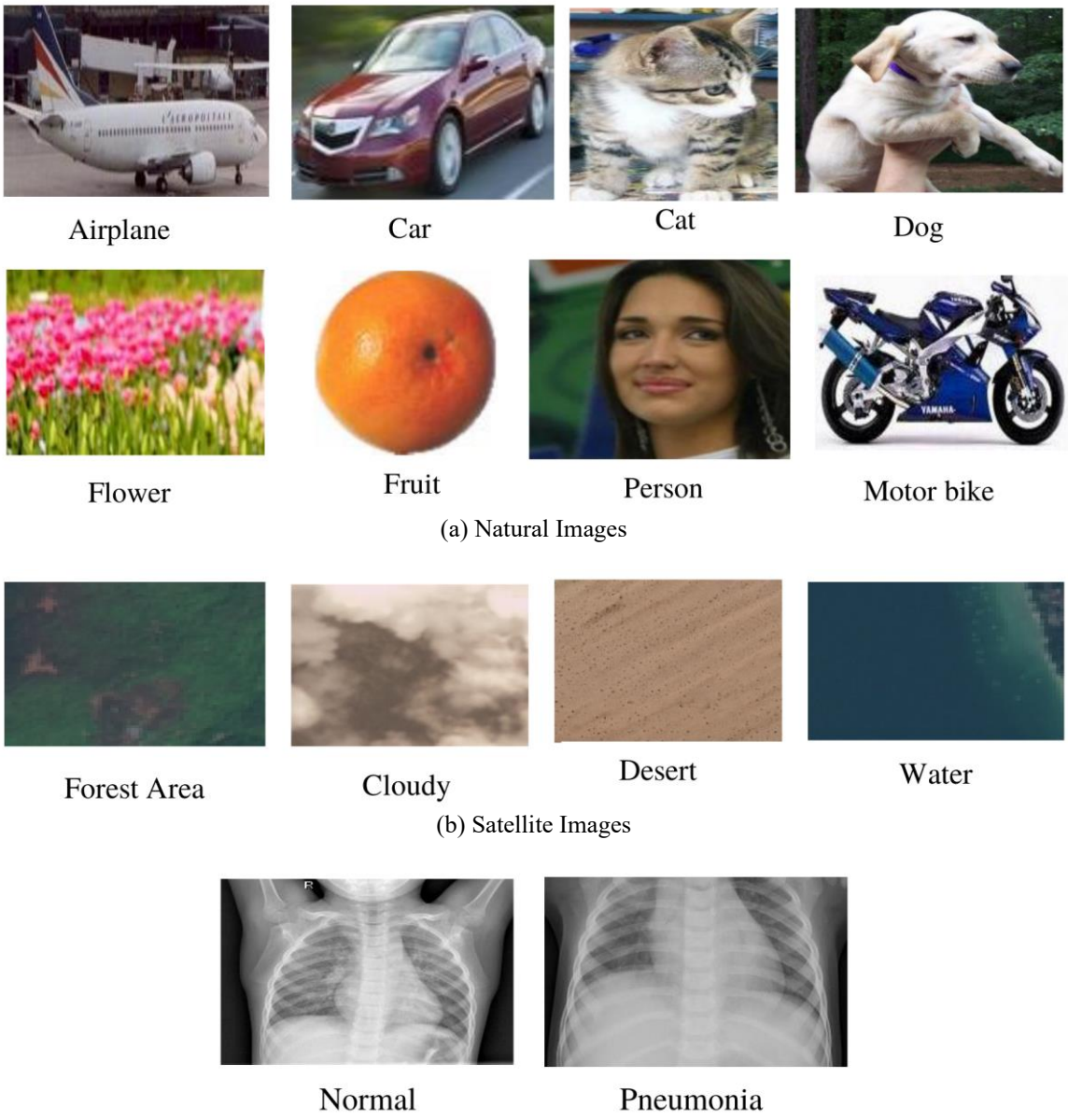
The Image Classification System implemented in this project is a user-friendly web application built using Python, Flask, TensorFlow, SQLite, HTML, CSS, JavaScript, and Bootstrap. It allows users to upload an image file, which is then preprocessed and classified to predict the class or category of the image. The prediction is performed using a trained convolutional neural network model, and the user is provided with the result together with the probability of the prediction. In addition to image prediction, the system allows users to manage categories, view statistics, and update their profiles, and it includes a live object detection feature based on YOLOv8. Sample images from each of the three domains covered by the system are shown in Fig. 1.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15837



(a) Natural Images

(b) Satellite Images

(c) Medical Images

Fig. 1 Sample Images from the Multi-Domain Dataset

1. Problem Statement

The exponential rise in the number of digital images has made manual classification a tedious, time-consuming, and error-prone task. Conventional image classification approaches suffer from being highly subjective, requiring extensive manual effort and providing inaccurate results for different types of images. There is a need for an automated system that can perform accurate classification of Natural, medical, and Satellite images using deep learning algorithms. This project addresses this problem by creating a web application that allows users to upload images and provides accurate classification results in a short time, and it further includes the feature of detecting objects in an image in real time.

2. Objectives

1. To design and develop a web-based image classification system that can classify Natural, Medical, and Satellite images accurately and efficiently.
2. To design and develop deep learning-based classification models that accurately predict image categories.
3. To develop a front-end interface where the user can upload images, view results, check the prediction history, and use the system comfortably and intuitively.

DOI: 10.17148/IJARCCE.2026.15837

4. To design and implement a feature for live object detection using a webcam or video stream and demonstrate effective use of the computer vision field.

3. Scope and Applications

The scope of the proposed system is the automated identification of Natural, medical, and Satellite images through a web interface. The system is designed to be easily scalable to include more categories of pictures, larger amounts of data, the ability to work in the cloud, and to process images in real time. Its principal application areas are the following.

- **Healthcare:** analysis of X-ray, MRI, CT scan and ultrasound images to identify pneumonia, fractures and other conditions such as cancer at early stages.
- **Environmental:** analysis of aerial or satellite pictures to study the Earth's surface and classify habitats into forests, waters, deserts and similar categories.
- **Object Recognition:** identifying and recognising natural objects such as humans, animals, transportation, flora and structures accurately and quickly.
- **Security:** identifying objects in photographs or surveillance video to spot irregularities and detect illegal actions, objects, persons or other suspicious activity.

The rest of the paper is organised as follows: Section II presents the related work and reviews existing methods for image classification and object detection. Section III describes the proposed methodology. Section IV describes the technology used, while Section V discusses the results. Section VI presents the conclusion and future work. Finally, Section VII includes the references.

II. RELATED WORK

Recent advances in computer vision have significantly improved automated image classification and real-time object detection. CNNs, transfer learning, lightweight architectures, and YOLO-based detectors have been widely investigated to improve accuracy while reducing computational complexity.

Sandler et al. [1] proposed **MobileNetV2**, which uses inverted residual blocks and linear bottlenecks to reduce parameters and computational cost while maintaining effective image classification and transfer-learning capability. Howard et al. [2] introduced **MobileNetV3**, incorporating neural architecture search, squeeze-and-excitation modules, and optimized activation functions to further improve the accuracy-efficiency trade-off. Krizhevsky et al. [3] demonstrated the effectiveness of deep CNNs through **AlexNet**, which employed convolutional layers, ReLU, dropout, and GPU computation for large-scale image classification. He et al. [4] proposed **ResNet**, using residual connections to overcome degradation in deeper networks and improve feature learning.

For object detection, Li et al. [5] developed **YOLOv6**, incorporating an optimized backbone and feature-fusion strategies for accurate real-time detection. Terven and Cordova-Esparza [6] reviewed YOLO architectures from YOLOv1 to YOLOv8 and YOLO-NAS, reporting progressive improvements in accuracy and inference efficiency. Khurana et al. [7] proposed **MOLO**, which combines MobileNet and YOLOv8 to achieve efficient object detection on resource-constrained devices. Gallagher and Oughton [8] surveyed YOLO-based multispectral object detection and highlighted recent developments, applications, datasets, and challenges.

Deep learning has also been applied to domain-specific detection and classification. Manolakis et al. [9] combined YOLOv5 with Depthwise SegNet for breast-lesion detection and segmentation, demonstrating the effectiveness of lightweight two-stage frameworks in medical imaging. Deng et al. [10] proposed a lightweight CNN for UAV image classification and showed that compact architectures can provide improved classification accuracy with efficient processing.

Although these studies demonstrate substantial progress in image classification and object detection, most focus on individual tasks or experimental model evaluation. Few integrate classification, real-time detection, user authentication, prediction history, and performance monitoring into a single accessible platform. The proposed system addresses this gap by integrating a CNN-based image classification module and YOLOv8-based real-time object detection within a Flask web application.

Some proposed models are implemented only in research and are not engineered to be used as web applications; most require end-users to have background knowledge in programming and therefore do not support general use. Others are designed to operate on particular datasets only, and thus lack flexibility in incorporating other categories of images. The proposed system addresses these gaps by combining a deep learning model with a deployed Flask web application offering user registration, image upload and prediction, prediction history, category management, a performance

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15837

dashboard and profile management, and by remaining flexible enough to adopt new datasets and advanced deep learning models in the future.

III. METHODOLOGY

The proposed Image Classification System classifies any image using a Deep Learning approach. The system is designed to facilitate the image classification process by employing a trained Convolutional Neural Network model and a Flask web application in order to provide an accurate, reliable, and convenient image classifier.

The application operates through a step-by-step process which commences with the user logging in to the system. Once logged in, the user can upload the desired image via the web interface. The uploaded image is then subjected to preprocessing steps such as resizing and normalisation, followed by the classification step performed by the CNN model. Upon successful classification, the application displays the image along with its corresponding label, and the result, together with the image URL, is stored in a database so that it can be retrieved whenever required.

1. Existing System

Existing image classification systems mainly rely on traditional machine learning techniques or standalone deep learning models. Many of these systems require manual feature extraction before classification, making the process time-consuming and less accurate for complex images. Several research-based models are designed only for experimental purposes and operate through programming environments instead of user-friendly web applications, and often lack important functionality such as secure user authentication. Many available applications focus only on generating prediction results without maintaining user records or providing an interactive interface, so users with limited technical knowledge find them difficult to operate.

2. Proposed System

The proposed Image Classification System is a Flask web application that uses a deep learning model to classify images. The application allows users to register, log in, upload images, and view the prediction results. The prediction is performed by a convolutional neural network trained using TensorFlow and Keras. The uploaded image is first preprocessed, and the model then predicts the image category with a high level of accuracy. Besides image prediction, the system provides history, category information, visualisation, user profile, and database features, and the interface built with HTML, CSS, Bootstrap, and JavaScript makes it easy to use and responsive.

a) Advantages of the Proposed System

- Automated image classification requiring the least manual effort for classifying images.
- Accurate prediction results using a trained deep learning model, with no manual feature extraction.
- A user-friendly graphical web interface for image classification.
- Secure user registration and authentication, with prediction history maintained for future reference.
- An efficient image preprocessing stage and a less time-consuming recognition process.
- Easily modified to include more classes and image datasets for better accuracy.
- Better scalability and maintainability, since the system is designed using a modular approach.
- Usable by many organisations for academic, research, medical, agricultural, industrial, and security purposes.

3. System Architecture

The Image Classification System is designed based on the client-server architecture. Users access the application remotely through a web browser; the system backend is built using the Flask framework, and the deep learning model handles the image classification process.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15837

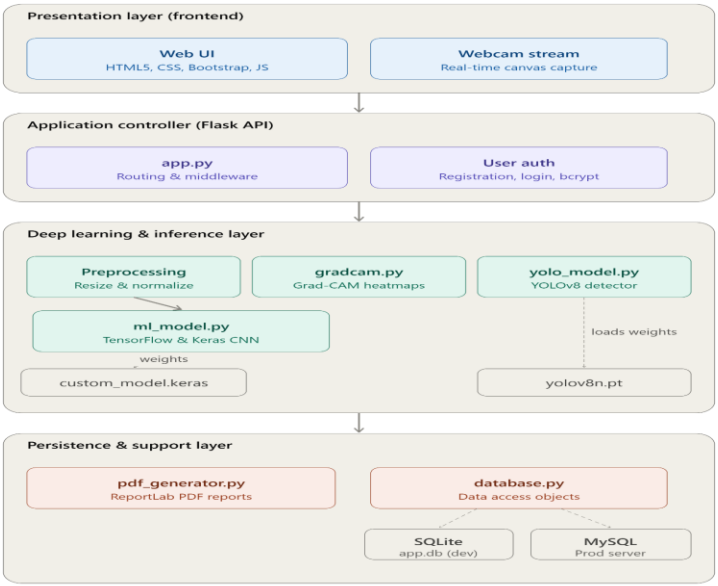


Fig. 2 System Architecture

The system works in the following manner: the user registers and logs in to the application and uploads the desired image through the dashboard; the Flask web server receives the request and the uploaded image; the image is preprocessed using OpenCV; the preprocessed image is input into the trained TensorFlow/Keras CNN model; the model classifies the image and returns the predicted category alongside the confidence score; the result is presented to the user and then stored in an SQLite database. From the dashboard, users can view their prediction history, categories, and personal profile.

The system is therefore designed using the following modules: the HTML, CSS, Bootstrap and JavaScript front end; the Flask web server; the image preprocessing module; the CNN prediction model; the SQLite database; and the user dashboard and history. Besides the simplicity of the design, this architecture supports easy maintainability, extensibility, and modularity.

4. Workflow

New users are navigated to the registration page, where name length, Gmail format, and password length are validated, while returning users are directed to the login screen where credentials are verified against the SQLite store. Inside the application, the user is greeted by an overview card displaying the total classifications, average confidence, and most frequently identified class. From the menu the user can reach Prediction, where the selected image is captured, resized and converted using PIL and processed by the model with file size, format and aspect ratio captured as metadata; History, which displays all previous predictions in chronological order and allows individual records to be deleted; Performance, which displays loss and accuracy plots and the validation confusion matrix; Live Feed, which initialises the camera and displays live video with YOLOv8 bounding boxes; Profile, where account details and passwords are updated; and Exit, which logs the user out.

a) Convolutional Neural Network (CNN)

Convolutional Neural Network is a type of deep learning algorithm used for image classification. CNN has the ability to automatically learn and detect patterns and features in image data without the need for manual feature detection and selection. It uses a layered approach: the process starts with applying convolution filters to an image to extract useful features such as edges, colour, shape, texture, and patterns. This layer is followed by a ReLU activation function to introduce non-linearity into the model. Global average pooling is then performed to downscale the feature matrices, and the output is fed into a fully connected layer followed by a softmax activation function, which predicts the target class of the input image.

Algorithm: CNN for image classification

1. Load the image dataset.
2. Resize the image to 224 × 224 pixels.
3. Normalise the pixel values.
4. Apply convolution filters to extract features from the image.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15837

5. Apply the Rectified Linear Unit (ReLU) activation function.
6. Perform max pooling to reduce the image size.
7. Flatten the pooled image.
8. Use fully connected layers to classify the image.
9. Use the softmax layer to predict the target label.
10. Display the final classified image with label and probability.

b) YOLOv8 Object Detection Model

YOLOv8 (You Only Look Once, Version 8) is an advanced deep learning model used for object detection in images, videos, and real-time video streams. Unlike other object detection models, YOLO processes images in grids, detecting objects in each grid and predicting their corresponding classes and locations in one pass. Developed by Ultralytics, it offers high accuracy with fast processing speed, has inherent real-time detection capability, and can simultaneously detect multiple objects in an image.

Algorithm: YOLOv8 for object detection

1. Upload the desired image.
2. Preprocess the image by resizing it.
3. Feed the image to the backbone of the YOLOv8 model.
4. Extract image features from the processed image.
5. Predict the bounding boxes of the detected objects.
6. Predict class labels with confidence scores.
7. Apply non-max suppression to remove duplicate boxes.
8. Draw the bounding boxes around the detected objects.
9. Display the final image with bounding boxes and confidence values.

IV. METHODS AND MATERIALS

1. Experimental Setup

The Image Classification System was designed and implemented on a Windows platform using the Python programming language and the Flask web framework. The hardware comprised an Intel Core i3/i5 processor with at least 4 GB RAM (preferably 8 GB), a 500 GB HDD/SSD to store the dataset, the trained model, and the project files, and a monitor with a resolution of 1366 × 768 pixels. The operating system was Windows 10/11, the application was coded in Python 3.x, and TensorFlow 2.x and Keras were used to build and train the CNN model. Visual Studio Code was used as the editor, SQLite as the database for user and prediction records, and HTML5, CSS3, Bootstrap, JavaScript, and Jinja2 templates for the front end.

2. Libraries and Frameworks

Flask: Flask is the lightweight Python web framework used to build the application. It handles routing, form parsing, session management and file uploads, and renders the prediction, history, categories, performance and profile pages.

TensorFlow and Keras: TensorFlow is the deep learning framework used to train the model, and Keras provides the high-level API used to build and train the CNN. Together, they produce the trained model that performs the classification at inference time.

OpenCV: OpenCV is used for image preprocessing and manipulation, including reading the uploaded file, resizing it to the model input dimensions, and converting it into the pixel arrays required for prediction.
NumPy: NumPy provides the numerical computation and array operations used to normalise and reshape the image arrays before they are passed to the model.

Ultralytics YOLOv8: The Ultralytics YOLOv8 platform provides the yolov8n.pt detection model used for the live object detection feature, drawing bounding boxes and confidence values over the camera stream.
SQLite: SQLite is the database used to store user account information and the prediction history that populates the dashboard and history pages.

HTML5, CSS3, Bootstrap and JavaScript: These are used to build the responsive front end through which users register, upload images and view results.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15837

Table 1. Tools and Technologies Used

Tool / Technology	Purpose
Python	Backend programming language
Flask	Web application framework
TensorFlow	Deep learning framework
Keras	Building and training the CNN model
OpenCV	Image preprocessing and manipulation
NumPy	Numerical computation
SQLite	Database management
HTML5	Web page structure
CSS3	User interface styling
Bootstrap	Responsive web design
JavaScript	Client interface scripting
Visual Studio Code	Source code editor
Git	Version control
Windows 10/11	Development environment

V. RESULTS AND DISCUSSION

1. Dataset Description

A multi-domain image classification dataset was gathered, comprising a diverse range of image categories, namely natural, satellite, medical and everyday images. The dataset consists of 12,783 RGB images of 14 classes distributed across three major groups (Natural, Medical and Satellite images). The images were sourced from open databases such as Kaggle, MS COCO and clinical MRI databases, and standardised into one common format. The dataset includes images with different sizes, colours, backgrounds and orientations, which allows the model to generalise the image features correctly and avoid overfitting, and it is divided into a training set used to train the CNN, a validation set used to evaluate performance and prevent overfitting, and a test set used to calculate the final performance score.

Table 2. Dataset Specifications

Parameter	Details
Dataset Name	Multi Domain Image Classification Dataset
Data Type	Image Dataset
Image Format	JPG / JPEG / PNG
Image Categories	Natural, Medical, Satellite
Source	Kaggle Datasets
Total Images	12,783 images
Image Size	Resized to 224 × 224 pixels
Preprocessing	Resizing, RGB normalisation, caching, prefetching

Table 3. Class Distribution of the Dataset

Group	Class	Count
Natural	Airplane	727
Natural	Car	968
Natural	Motor bike	788
Natural	Cat	885
Natural	Dog	702
Natural	Flower	843
Natural	Fruit	1,000
Natural	Person	986
Medical	Chest X-ray Normal	1,341
Medical	Pneumonia	1,224
Satellite	Cloudy	1,500
Satellite	Desert	1,131
Satellite	Green Area	1,500
Satellite	Water	1,500

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15837

2. Training and Testing

The deep learning model was trained on the labelled image dataset, which was split into training, validation, and testing sets to evaluate the model performance correctly. During the training phase, the CNN learned image features such as edges, textures, colours, and patterns using multiple convolutional layers, and performance was validated using the validation set to avoid overfitting. The testing process was completed in several steps: the user accesses the prediction page and uploads the desired image; the uploaded image is normalised and resized; the preprocessed image is input into the CNN model; the model returns the most probable category; the predicted category is displayed with its level of confidence; and the prediction is saved in the SQLite database. Several test images from different categories were used to check that the model predicted the categories correctly, and the prediction process took only a few seconds, showing that the CNN model performed efficiently.

3. Results and Analysis

The developed Image Classification System was able to classify uploaded images into their respective categories using the trained CNN model. Using Flask together with TensorFlow and Keras enabled real-time prediction through a user-friendly web platform. The system correctly performed user registration, user login, dashboard navigation, image uploading, image preprocessing, image classification, prediction viewing, prediction history, category display, profile management, and logout.

The CNN model was able to predict the images with good accuracy on the trained dataset. The image preprocessing step helped to enhance the prediction results, since the images were standardised before classification. The prediction history feature allowed the user to view past predicted images, and the dashboard gave an overview of the application modules. The system therefore performed accurate and efficient image classification with ease and reliability.

Table 4 Predicted Classes and Confidence Level

Class	Confidence
Green Area	98.3%
Desert	100.0%
Water	99.9%
Flower	100.0%
Person	97.9%
Dog	85.1%
Cat	100.0%
Car	100.0%
Fruit	100.0%
Pneumonia	98.8%
Normal chest X-ray	99.1%

Table 5. Reported Performance of Existing Deep Learning Methods and the Proposed System

Method	Task	Dataset / Benchmark	Reported Result
AlexNet [3]	Image Classification	ImageNet	63.3% Top-1 Accuracy
MobileNetV2 [1]	Image Classification	ImageNet	71.88% Top-1 Accuracy
MobileNetV3-Large [2]	Image Classification	ImageNet	75.2% Top-1 Accuracy
ResNet-50 [4]	Image Classification	ImageNet	76.13% Top-1 Accuracy
YOLOv6-N [5]	Object Detection	COCO	35.9% AP
YOLOv6-S [5]	Object Detection	COCO	43.5% AP
YOLOv8n	Object Detection	COCO	37.3% mAP@0.5:0.95
Proposed CNN	**Image Classification**	**Proposed 14-class dataset**	**96.9% Validation Accuracy**

Existing CNN architectures such as AlexNet, MobileNetV2, MobileNetV3, and ResNet have demonstrated strong image-classification performance on ImageNet, while YOLO-based models provide efficient real-time object detection on COCO. The proposed system differs by integrating CNN-based multi-domain image classification with YOLOv8-based object detection in a web-based application. Direct numerical comparison should be interpreted cautiously because the existing methods and the proposed system are evaluated on different datasets.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

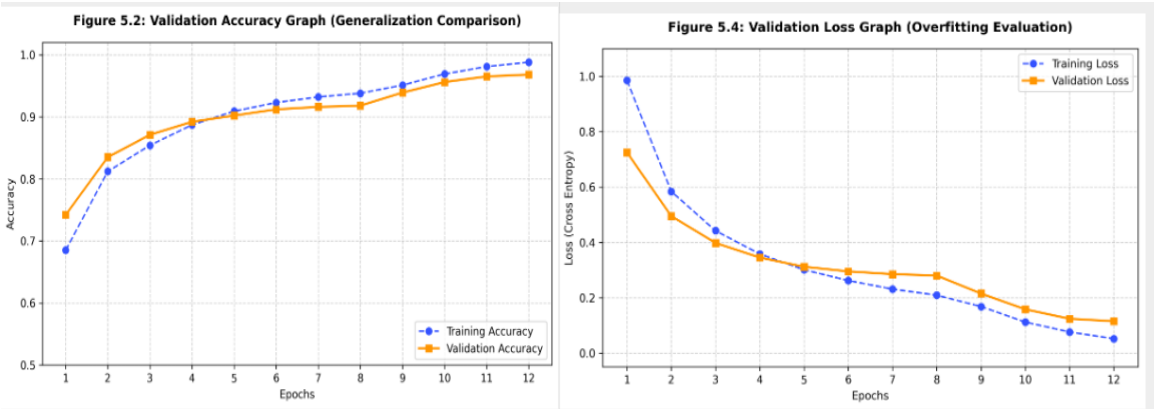
International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15837

4. Graphs and Charts

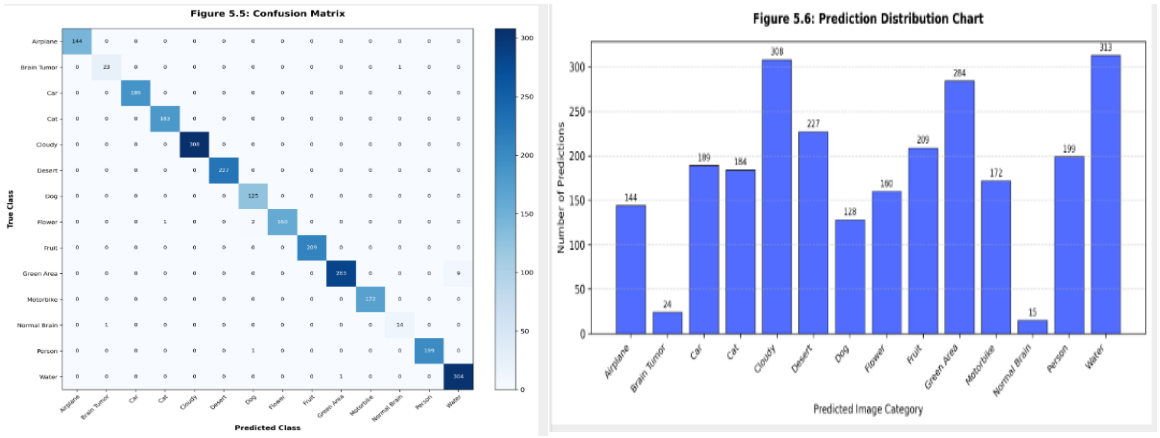
The performance of the CNN model is presented through graphs visualised from the training and testing processes.



(a) Validation Accuracy Graph **(b) Validation Loss Graph**

Fig. 3 Training and Validation Performance of the CNN Model: (a) Accuracy and (b) Loss

The training and validation accuracy curves show that the CNN performance improves progressively across epochs, with both curves converging toward high accuracy. The training and validation loss steadily decrease, indicating effective learning with no significant overfitting.



(a) Confusion Matrix **(b) Prediction Distribution Chart**

Fig. 4 Confusion Matrix and Prediction Distribution of the CNN Model: (a) Confusion Matrix and (b) Prediction Distribution Chart

The confusion matrix shows the distribution of correct and incorrect predictions across the different image classes, with strong diagonal values indicating accurate classification.

The prediction distribution chart illustrates the number of images predicted for each class, highlighting the variation in classification outcomes across categories.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15837

Table 6. Comparison of the CNN and YOLOv8 Models

Parameter	Proposed CNN (MobileNetV2)	YOLOv8n
Task	Multi-class image classification	Real-time object detection
Backbone	MobileNetV2, transfer learning	CSPDarknet with C2f modules
Input resolution	224 × 224	640 × 640
Training in this work	Trained for 12 epochs on the 14-class dataset	Used with pre-trained YOLOv8n.pt weights
Final training accuracy	0.990	Not applicable
Final validation accuracy	0.969	Not applicable
Final training loss	0.050	Not applicable
Final validation loss	0.115	Not applicable
Reported benchmark result	Not applicable	37.3% mAP@0.5:0.95 on COCO
Average confidence observed	94.6% over the prediction history	0.91 to 0.97 on live webcam frames

Table 6 compares the two models used in the system. The CNN was trained on the multi-domain dataset for the classification task, and its accuracy and loss values are those given in Table 6. YOLOv8n was used with its pre-trained COCO weights for the live detection feature and was not retrained in this work, so its accuracy and loss values are reported as the published benchmark result rather than as values measured here. The two models therefore complement each other: the CNN assigns a single category to an uploaded image, while YOLOv8n locates and labels multiple objects in the camera stream.

5. Snapshots of the web application

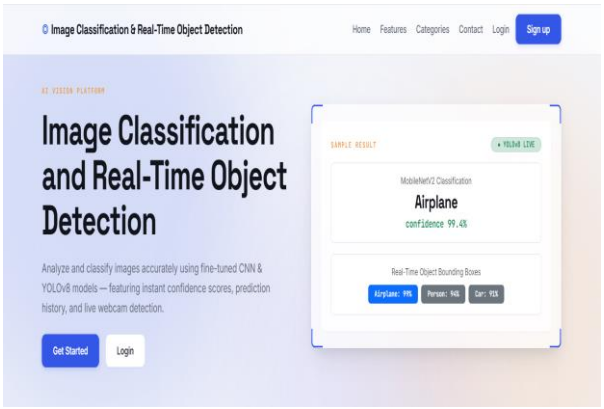


Fig. 5 Home Page

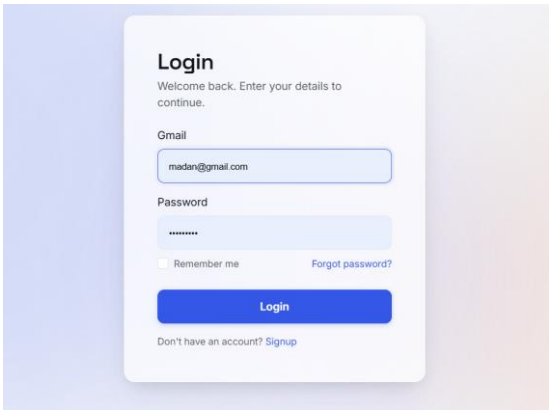


Fig. 6 User Login Page

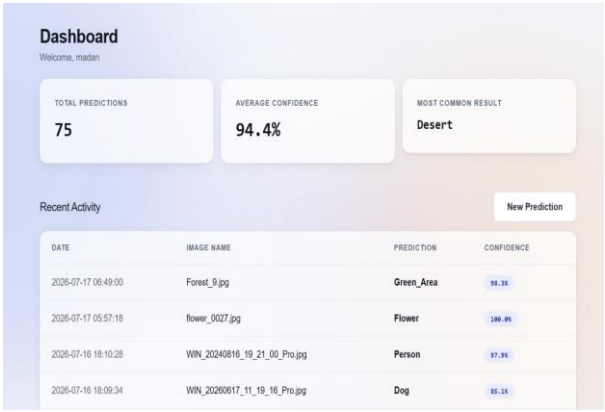


Fig. 7 Dashboard

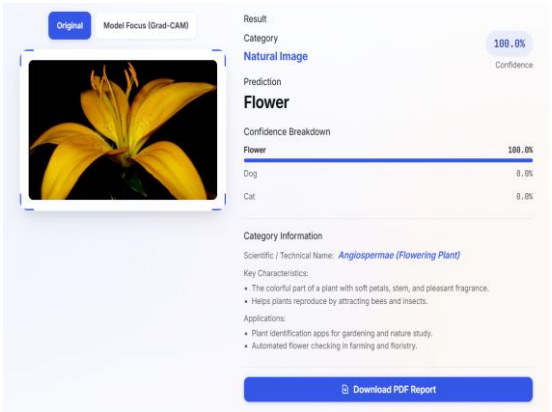


Fig. 8 Prediction Results

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15837

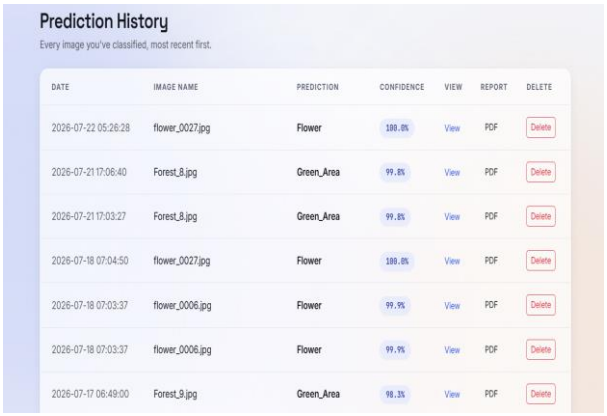


Fig. 9 Prediction History

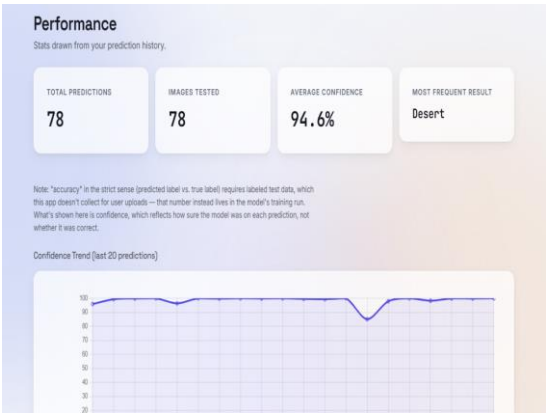


Fig. 10 Performance Page

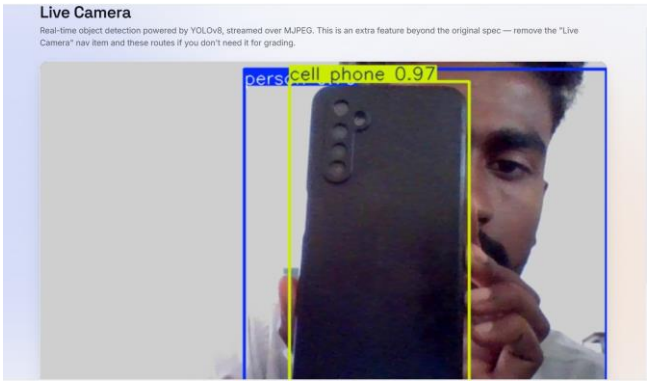


Fig. 11 Live Object Detection Interface

The Home Page is the landing page of the system, where the main features are presented together with the Login and Register options. The User Login Page is where registered users access the application, and the Dashboard is the main screen from which all features are reachable, including image prediction, prediction history, categories, performance, and profile. The Prediction Results page displays the predicted image together with the predicted label and level of accuracy, while the Prediction History page shows all predictions of the logged-in user with the class, confidence, and time of prediction. The Performance Page displays the accuracy of the model on the test images, and the Live Object Detection interface displays the camera feed with YOLOv8 identifying objects in real time.

VI. CONCLUSION

The proposed system successfully integrates CNN-based image classification and YOLOv8-based real-time object detection into a user-friendly Flask web application. The system classifies multi-domain images and provides prediction confidence, history management, and live object detection, demonstrating its effectiveness for practical computer vision applications.

Future work will focus on expanding the dataset and categories, exploring advanced models such as EfficientNet, ResNet, DenseNet, and Vision Transformers, and improving deployment through cloud and mobile platforms. Further enhancements include Grad-CAM-based explainability, multilingual support, stronger security, multiple-image processing, object tracking, and custom YOLOv8 model training.

REFERENCES

[1]. M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 4510–4520. [Online]. Available: https://arxiv.org/abs/1801.04381

[2]. A. Howard et al., "Searching for MobileNetV3," in Proc. IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 1314–1324. [Online]. Available: https://arxiv.org/abs/1905.02244

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15837

[3]. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Communications of the ACM, vol. 60, no. 6, pp. 84–90, 2017, doi: 10.1145/3065386.

[4]. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778. [Online]. Available: https://arxiv.org/abs/1512.03385

[5]. C. Li et al., "YOLOv6: A single-stage object detection framework for industrial applications," arXiv preprint arXiv:2209.02976, 2022. [Online]. Available: https://arxiv.org/abs/2209.02976

[6]. J. R. Terven and D. M. Cordova-Esparza, "A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8 and YOLO-NAS," Machine Learning and Knowledge Extraction, vol. 5, no. 4, pp. 1680–1716, 2023, doi: 10.3390/make5040083.

[7]. K. Khurana, P. Sonsare, D. Borkar, H. Thakkar, and O. Bhagat, "MOLO: A hybrid approach using MobileNet and YOLO for object detection on resource-constrained devices," Discover Artificial Intelligence, 2025, doi: 10.1007/s44163-025-00398-3.

[8]. J. E. Gallagher and E. J. Oughton, "Surveying You Only Look Once (YOLO) multispectral object detection advancements, applications and challenges," IEEE Access, vol. 13, 2025. [Online]. Available: https://arxiv.org/abs/2409.12977

[9]. D. Manolakis, P. Bizopoulos, A. Lalas, K. Votis, and D. Tzovaras, "A two-stage lightweight deep learning framework for mass detection and segmentation in mammograms using YOLOv5 and Depthwise SegNet," Journal of Imaging Informatics in Medicine, 2025, doi: 10.1007/s10278-025-01471-0.

[10]. X. Deng et al., "A lightweight CNN model for UAV-based image classification," Soft Computing, vol. 29, pp. 2363–2378, 2025, doi: 10.1007/s00500-025-10512-3.

