Ingénierie des Systèmes d'Information
Vol. 30, No. 3, March, 2025, pp. 675-686
Journal homepage: http://iieta.org/journals/isi

Improving Objects Detection in Video with a Proposed 1D CNN Model

Farah Hussein M. Jawad[1,2*], May A. Salih[1]

[1] Department of Software, College of Information Technology, University of Babylon, Babil 51001, Iraq
[2] Department of Business Administration, College of Business and Economics, University of Babylon, Babil 51001, Iraq

Corresponding Author Email: Bus.farah.hus@uobabylon.edu.iq

Copyright: ©2025 The authors. This article is published by IIETA and is licensed under the CC BY 4.0 license (http://creativecommons.org/licenses/by/4.0/).

https://doi.org/10.18280/isi.300312

Received: 4 November 2024
Revised: 24 February 2025
Accepted: 12 March 2025
Available online: 31 March 2025

Keywords:
object detection, COCO dataset, convolutional neural network (CNN), principal component analysis (PCA), local binary pattern histogram (LBPH)

ABSTRACT

Object detection in video is one of the main tasks of computer vision, with applications ranging from surveillance to autonomous driving. This research aims to enhance object detection in video sequences using a convolutional neural network (CNN) model based on the COCO dataset. The study introduces an innovative 1D CNN design that leverages spatial and temporal information from successive frames to achieve accurate detection and recognition of objects. The proposed system consists of three phases: the preprocessing phase, the feature extraction phase by using two techniques: principal component analysis (PCA), and local binary pattern histogram (LBPH), and the detection phase based on the proposed object-CNN model. Experimental results indicate that the proposed model achieves a detection accuracy of 99.74%, outperforming existing state-of-the-art methods. This research significantly advances object detection capabilities in dynamic environments such as enhancing real-time surveillance systems or improving autonomous vehicle navigation.

1. INTRODUCTION

Object detection is a fundamental computer vision problem, can provide vital information for semantic comprehension of pictures and videos, and is connected to numerous applications, including image categorization, human behavior analysis, face recognition, and autonomous driving [1]. Meanwhile, advances in neural networks and associated learning systems will lead to the development of neural network algorithms, as well as significant influences on object identification approaches that may be called learning systems [2]. However, due to significant changes in perspectives, postures, occlusions, and lighting conditions, it is challenging to achieve a faultless object field that has received a great deal of interest in recent years [3].

The ability to accurately identify and localize objects of interest within video frames is fundamental for understanding the visual content of videos and enabling intelligent decision-making systems. The capability of CNN to learn hierarchical features directly from raw pixel data makes them increasingly popular in handling object detection tasks. Building on the recent success of CNNs in video analysis [4], there is therefore growing interest in the development of techniques that extend such methods to video data where the temporal dimension introduces further complexity and challenges.

This paper applies object detection utilizing the COCO dataset. The COCO dataset is chosen for its diverse and extensive collection of labeled images, providing comprehensive coverage of various object categories in real-world scenarios. Its rich annotations are ideal for training and testing an object detection model, particularly in dense and dynamic scenes [5].

The model architecture includes feature extraction convolutional layers, spatial downsampling pooling layers, and detection of fully connected layers. By evaluating our model, we aim to demonstrate its effectiveness in detecting objects in video sequences with varying complexities.

The main contributions of this study are the reduction of detection time while enhancing the performance of object detection technology in scenarios involving small objects and dense scenes, thereby offering more precise and dependable detection services across diverse application domains.

The structure of this paper is as follows: Section 2 provides related works on object detection utilizing deep learning techniques. Section 3 explains the methodology of the proposed system. Section 4 provides experimental results and discussion. Section 5 concludes the paper with key findings and limitations.

2. RELATED WORKS

Thanks to the fast advances in deep learning over the past decade, computer vision has significantly progressed, particularly through the development of deep CNNs. CNN is built using key components that work together to process and analyze data, typically images. The Convolution Layer is the core, using filters to extract features like edges or patterns from the input. The Pooling Layer reduces the size of feature maps for efficiency, with MaxPooling specifically selecting the most important values. To introduce non-linearity, the LeakyReLU activation function allows small gradients for

negative values, solving issues like "dying neurons." The Flatten Layer reshapes the multi-dimensional data into a 1D vector, preparing it for the Dense Layer, which performs high-level reasoning (e.g., classification) by connecting all neurons. Together, these components allow CNNs to effectively learn and recognize patterns in data.

These networks have become the standard approach for tackling the object detection problem, which involves both object location and recognition. Within this domain, the most advanced models are categorized into Two-Stage and Single-Stage Detectors.

Each method has its merits and demerits in investigating Two-Stage and Single-Stage Object Detectors, which are used in each method. Two-stage detectors often have higher accuracies, as seen by Cascade R-CNN and EfficientDet, suitable for those applications where high accuracy is highly valued. However, they often rely on the initial region proposals' quality and higher computation might be a limiting factor, especially in smaller datasets.

On the other hand, one-stage detectors represent the YOLO family, where speed is its strong point. These models are quite fast in processing images, but this also comes at the price of accuracy. In difficult cases involving small or occluded objects, performance usually suffers. As much as this model is pretty good at quick inference, it may have very disparate performance depending on the dataset or context.

Ultimately, the decision between using a Two-Stage or Single-Stage Detector should be based on the specific needs of the application whether that's prioritizing accuracy or requiring quick responses. Looking ahead, it will be important for researchers to focus on improving the robustness of these models, tackling their limitations, and considering innovative approaches that combine the strengths of their architectures. By doing so, we can continue to advance the capabilities of object detection systems to meet the demands of diverse real-world applications. Table 1 summarizes related works, drawing on some of the most important approaches, methodologies, and findings.

Table 1. Summary of related works

Ref.	Year	Technique	Dataset	Measures	Advantages	Disadvantages
Two-Stage Detectors						
[6]	2019	ResNet-50	ImageNet	Accuracy 75%, Precision 76%, Recall 72%	Very deep network with residual connections	A larger model size requires substantial computation
[7]	2020	MobileNetV2	ImageNet	Accuracy 71.8%, Precision 71.2%, Recall 68.5%	Lightweight, suitable for mobile deployment	Lower accuracy compared to larger models
[8]	2023	Faster R-CNN	COCO	Accuracy 87.7%, Precision 84.5%, Recall 85.0%	good accuracy for object detection	Slower than one-stage models
[9]	2023	Cascade R-CNN	COCO	Accuracy 88.2%	Multi-stage detection pipeline improves accuracy	Increased computational cost
[10]	2023	CNN	CDnet	Precision=90%	Good precision; suitable for specific applications with limited object types	This study couldn't detect various object types in different datasets; the training process was also complex
Single-Stage-Detectors						
[11]	2024	YOLOv5	Real-world object detection datasets	60.9% mAP	Low detection accuracy	May require significant computational resources; performance might drop in real-time applications
[12]	2020	EfficientDet-D7	COCO	Accuracy 89.5%, Precision 84.0%, Recall 82.0%	Balances accuracy and efficiency	The performance of EfficientDet is sensitive to hyperparameter choices and must be optimized for specific contexts
[13]	2024	YOLOv8	COCO	Accuracy=94%, Precision=93%, Recall=94%, F1 Score=0.93	Training complexity; requires large datasets for effective performance	High accuracy; improved architecture over previous YOLO versions; very efficient in real-time applications
[14]	2023	YOLOv9	COCO	Accuracy=93.5%, Precision=91.0%, Recall=92.0%,	High precision in detection	Performance varies in cluttered scenes
[15]	2024	YOLOv10	COCO	Accuracy=91.0%, Precision=90.0%, Recall=92.0%,	Advanced improvements over prior YOLO versions	Still in ongoing evaluation for real-world applications
Both of Two-Stage and Single-Stage						
[16]	2019	YOLO, CNN, Fast CNN	Various object detection datasets	Comparison of detection speeds and bounding box accuracy	Fast processing; effective for different applications; good bounding box accuracy	Limited to simulated environments
[17]	2020	R-CNN, YOLO v3	Collected dataset	Accuracy=79%	Combines strengths of R-CNN and YOLO; effective for diverse object detection tasks	It is not appropriate for real-time applications due to its considerable computing capabilities, which necessitate parallel computing
[18]	2022	YOLOv3,	Bad weather	YOLOv4: 72% mAP,	Effective in challenging	Only tested under specific weather

		YOLOv4, Faster R-CNN	and low-light datasets	63% recall at 40,000 iterations	conditions; performs well in low-light scenarios	conditions, limiting generalizability
[19]	2024	YOLO, Fast-RCNN	Object detection datasets	YOLO: 63.4 mAP, 300x faster inference time	Fast inference; effective for real-time applications; suitable for general object detection tasks	Performance may vary based on the dataset
[20]	2024	R-CNN, SSD, YOLO	Chess Piece	Best accuracy for R-CNN=78%	High accuracy for R-CNN; SSD and YOLO provide speed advantages; effective for specific applications like chess piece detection	The implementation bottleneck is caused by the use of an independent region generator

2.1 Studies of two-stage detectors

Region proposal and classification stages make up the two stages of two-stage frameworks used in the detection process. First, by employing reference boxes (anchors), these models suggest several potential objects, or regions of interest (RoI). Refined localization and classification of the ideas are the outcomes of the second step.

This model excels in feature extraction, making it a popular backbone for various applications [6]. Its architecture allows for training very deep networks, which can capture complex patterns in data, but this comes at the cost of increased computational requirements.

MobileNetV2 was designed for efficiency, making it ideal for mobile and embedded systems. Its architecture enables faster inference times while maintaining reasonable accuracy, though it sacrifices some performance compared to larger models [7].

A two-stage detection process significantly enhances accuracy, particularly in complex scenes. However, this comes at the cost of speed, making it less suitable for real-time applications compared to one-stage detectors [8].

Several multi-stage mechanisms were introduced to improve accuracy from varied scale changes of objects in 2023 [9]. Although these mechanisms enhance the results, they may imply increased computational burdens that are deterring factors towards a limited-resource deployment platform.

A deep learning-based system for real-time object identification was suggested in 2023 [10]. It focused on the classification and detection of stationary and moving objects. A CNN with a Softmax classifier was used, which showed an average precision of 90% for three video sequences of the CDnet database, proving its efficiency in real-time object identification.

2.2 Studies of single-stage detectors

On the other hand, one-stage detectors rely on a single feed-forward fully convolutional network that offers both object categorization and bounding boxes. The Single Shot MultiBox Detector (SSD) and YOLO (You Only Look Once) were among the first to suggest a single, unified structure that does not require per-proposal calculation.

The application of YOLOv5 in object detection was investigated, focusing on its architectural enhancements, functionalities, training process, and transfer learning techniques [11]. The results indicate that YOLOv5 is an essential technique in computer vision, with a mean Average Precision of 60.9%. Its framework enhances mean average accuracy, computational flexibility, and dependability, making it suitable for real-world applications in computer vision and video processing.

EfficientDet-D7 provides a good trade-off between accuracy and computational efficiency, as it is an all-around network across different platforms. However, its performance varies in some applications and requires further tuning for optimal results [12].

YOLOv4 was proposed in 2020, extending the YOLO architecture with new features and upgrades, such as the CSPDarknet53 backbone and PANet path aggregation. It achieved up to 89.8% mAP on the COCO dataset, balancing speed and accuracy effectively.

YOLOv8 was introduced in 2024 [13], achieving a mean Average Precision of 95.4%, with an accuracy of 94%, precision of 93%, recall of 94%, and an F1 score of 0.93. It addresses both speed and accuracy, making it suitable for applications requiring fast decision-making, such as autonomous vehicles and surveillance systems. However, training YOLOv8 is computationally intensive and requires large datasets.

YOLOv9 was developed in 2024 to be suitable for fast and accurate detection of distinct objects [14]. However, its performance degrades in cluttered scenes and requires further refinement for complex environments.

Further development related to the YOLO series was introduced in 2024, incorporating more sophisticated features to enhance detection performance. This version is still under evaluation and requires additional benchmarking to prove its efficiency in more general scenarios [15].

2.3 Studies used both single-stage and two-stage detectors

The YOLO technique was compared to other object detection algorithms, revealing several advantages. Unlike other methods, YOLO uses neural networks to identify bounding boxes and class probabilities, allowing it to view the entire image at once and thus detect objects much faster than methods like CNN and Fast CNN [16].

Object detection approaches were analyzed using two categories: one-stage detectors (e.g., YOLO v1, v2, v3, and SSD) and two-stage detectors (e.g., RCNN, Fast RCNN, and Faster R-CNN). One-stage detectors prioritize speed, while two-stage detectors focus on high accuracy. The YOLO v3-Tiny model achieved an accuracy rate of 79%, outperforming previous approaches [17].

The performance of YOLOv3, YOLOv4, and Faster R-CNN was compared for object detection in bad weather and low light conditions [18]. YOLOv4 achieved the highest results with 72% mAP and 63% recall at 40,000 iterations, outperforming YOLOv3 and Faster R-CNN.

Object detector performance was evaluated based on detection accuracy and inference time. Two-stage detectors generally outperform single-stage detectors in terms of accuracy, while single-stage detectors offer better inference times. The YOLO architecture has significantly improved detection accuracy, sometimes surpassing two-stage detectors. For example, YOLO models achieve detection accuracies of 63.4% and 70% for YOLO and Fast-RCNN, respectively, with

YOLO having around 300 times faster inference time [19].

Three object detection techniques were implemented: Faster R-CNN, R-CNN, and YOLO. The study aimed to find the optimal trade-off between feature extraction and accuracy. In this context, R-CNN achieved superior accuracy compared to single-stage detectors like YOLO or SSD, with an accuracy rate of 78% [20].

3. THE PROPOSED MODEL ARCHITECTURE

In this work, deep one-dimension CNN is integrated with techniques of feature extraction, PCA, and LBPH to extend the recognition ability of objects for detection tasks. COCO is applied in this proposed approach for object detection in the model, as well as captioning tasks on a given image. Its rich annotations and diverse set of object categories make it an ideal choice for training and evaluating the proposed architecture. First, the dataset spilled into two sets training set 70% and testing set 30%. Then, the dataset is processed by using some preprocessing methods (Converting RGB to Gray, Gaussian blur, Histogram Equalization, and image resizing). The second stage is the feature extraction techniques (PCA and LBPH).

The third stage was the detection phase which was accomplished by utilizing the suggested 1D-CNN model. This model forms the backbone of the architecture for its exceptional capability of automatically learning hierarchical features from input data, particularly well-suited for image processing tasks like object detection. Figure 1 presents the suggested model architecture The next sections will explain these phases in detail.

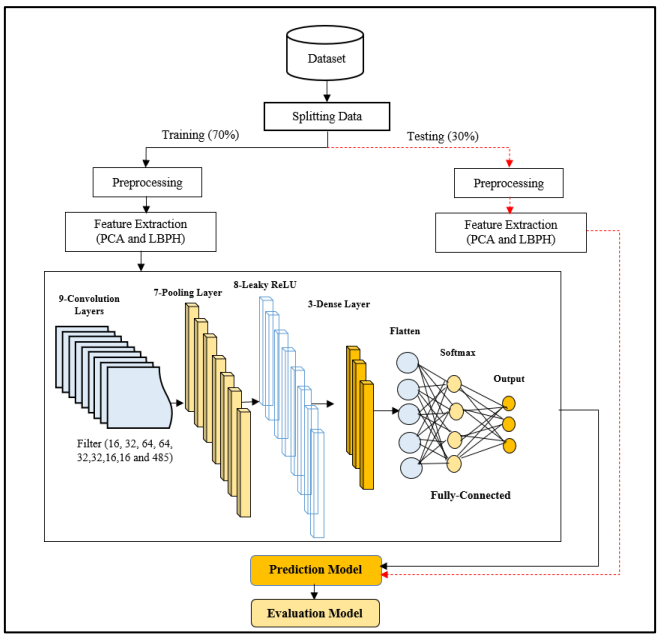

Figure 1. Diagram of the suggested model architecture

3.1 COCO dataset description

COCO is considered a benchmark standard in the computer vision community; it allows for the evaluation of object detection algorithms and their comparison. Headings, or heads, Large-scale object recognition, segmentation, key-point detection, and captioning are all comprised in the MS COCO (Microsoft Common Objects in Context) dataset Figure 2 shows some samples of this dataset. There are 328K images in the dataset 2014 saw the publication of the MS COCO dataset's first version. It has 164K images divided into sets for testing (41K), validation (41K), and training (83K). A fresh test set of 81K photographs, comprising 40K new images plus all of the test images from prior releases, was made available in 2015. In 2017, the training/validation split was restructured from 83K/41K to 118K/5K depending on input from the community. The annotations and images are the same in the new split. The 2017 test set is a subset of the 41K images from the 2015 test set. A new 123K image unannotated dataset is also included in the 2017 version [5]. This dataset is available at: (https://www.kaggle.com/datasets/sabahesaraki/2017-2017).

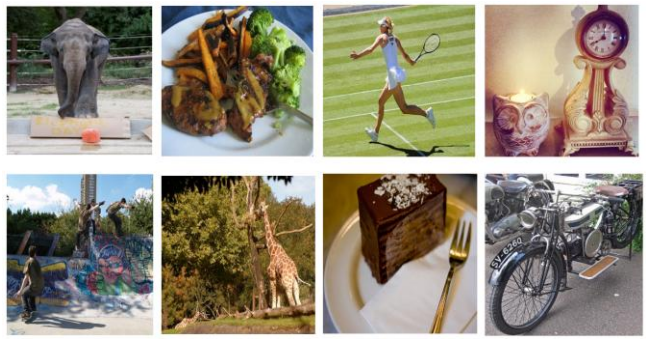

Figure 2. Samples of MS COCO dataset

Category composition:

The COCO dataset covers a diverse set of 80 fully differentiated object categories that represent a huge variety of common objects and living beings that one might find in the real world. Instances include forms of transportation, like cars, bicycles, motorcycles, and airplanes, but also traffic-related ones like traffic lights and stop signs.

Apart from vehicles, the dataset contains a wide array of animals, including domesticated ones like cats and dogs, and also farm and wild ones like cows, elephants, and giraffes. Again, personal things and accessories also include items like backpacks, handbags, frisbees, tennis rackets, and skis.

The dataset does not leave out food items, which include a variety of fruits such as bananas and apples, and prepared foods like pizzas, sandwiches, and cakes. Several household and furniture items are included: chairs, tables, and electronic devices such as TVs and laptops.

These categories extend to include everything in everyday life, such as forks, knives, spoons, different containers, household paraphernalia, bottles, and vases. With the broadness of these categories, there is a great avenue for the training of the model on how to conduct object detection and recognition exercises to identify so many objects that come about in real-world settings.

The dataset has annotations:

The COCO dataset is renowned for its high-quality annotations. The key aspects of annotation quality are:

• Object detection: 80 object types are represented by bounding boxes and per-instance segmentation masks; captioning: natural language explanations of the images,

• Identification of key points: including more than 200,000 images and 250,000 human examples with 17 possible key points (e.g., left eye, nose, right hip, right ankle); segmentation of stuff photographs: 12 stuff categories (e.g., grass, wall, sky) are segmented per pixel using segmentation masks.

• Panoptic: complete scene segmentation, comprising 12 stuff categories (grass, sky, road) and 80 object classes (person, bicycle, elephant),

• Dense Pose: over 39,000 images and 56,000 human instances have been annotated; each identified individual has a mapping between their body-corresponding picture pixels and a template 3D model, as well as an instance ID. We call this a thick stance. The public can only view the annotations for the training and validation images.

3.2 Dataset splitting

Data splitting is a technique utilized to validate models by dividing a dataset into two sets: testing and training. The testing set is utilized to validate the models fitted using the training set, allowing for comparison without overfitting [21]. Random subsampling is a popular method for data splitting, with a commonly used ratio of 80:20, which designates 80% for training and 20% for testing. Other ratios like 70:30, 60:40, and 50:50 are also used. The 80:20 split is based on the Pareto principle, but it is a practitioner-only guideline. The best or optimal ratio for a given dataset is not well defined, and the Pareto theory serves as a foundation for this method [22]. In this system, we used 70% as a training set and 30% as a testing set. It is a regularly used ratio, the 70-30 split where 70% of the data is employed for training and 30% for testing is not rigid and can alter depending on various variables, including the size of the dataset, the complexity of the model, and the particular job at hand. Here is a discussion about why the 70-30 split is commonly used:

- *Sufficient Training Data:* 70% of the data is used for training and this is ample. It captures enough of the core patterns and relationships in the data which gives a well-trained model.
- *Adequate Testing Data:* 30% of the data is kept as testing data to ensure there is enough data for unseen data testing. Too little data in testing data leads to over-fitting performances while too much may starve the training data, therefore weakening the ability to learn.
- *Balanced Training and Testing:* the 70/30 split is balanced in that one gets enough data for training, plus a good evaluation of generalization performance. It allows the model to learn from a substantial amount of data while leaving a large enough chunk to evaluate how well the model has learned to generalize from the examples.
- *Statistical Significance:* The 70/30 split often gives a statistically significant evaluation of the model's performance. With adequate data in both the training and testing sets, the resulting performance measures are likely to be accurate indications of the model's genuine capabilities.

3.3 Data preprocessing

Data preparation is essential for accurate data processing. Considering the inherent difficulty of operations for building and data quality constraints, this step is crucial for developing operational analysis methods [23]. Data preprocessing is a set of procedures for improving raw data quality, including outlier removal and imputation of missing values [24]. The preprocessing step, which includes techniques for segmentation, color conversion, image enhancement, and scaling, aims to turn data into a format that can be handled more rapidly and effectively. Figure 3 depicts these processes in detail which are discussed in four steps:

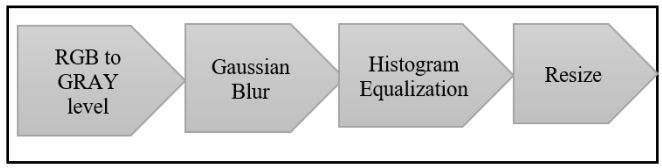

Figure 3. Pre-processing phase

3.3.1 Converting RGB to gray

The input image undergoes a color transformation from RGB to grayscale to reduce data usage as shown in Figure 4. Greyscale images have only one channel, unlike RGB images which require three channels [25]. This conversion increases processing speed. The grayscale image intensity is recorded as an 8-bit integer, resulting in 256 distinct grey spectrums ranging from white to black. This process reduces the amount of data used for image representation. The procedure of grayscale image transformation is shown in Eq. (1):

$$\text{Gray Image} = (0.21\ \text{R} + 0.72\ \text{G} + 0.07\ \text{B}) \quad (1)$$

Because people see green the most, the luminosity equation accommodates this by assigning green (G) the highest weight. RGB to grayscale conversions in object detection models simplify information because colors are perhaps not that important for the detection of objects. Grayscale usually focuses on structural and intensity details such as edges and shapes, which are often the critical features for object detection. By removing the complication of color from the model, the processing of images becomes much faster with less computational burden and is very necessary for video sequences of real-world performance. It also contributes to a reduction in noise due to color variances that are irrelevant in object localization and detection [26].

Figure 4. From RGB to grayscale image

3.3.2 Image Blurring using Gaussian Blur

This blurring method produces a smooth blur that resembles seeing a picture through a transparent screen. Gaussian smoothing is frequently employed as a preprocessing step in computer vision algorithms to improve visual structures of various sizes. Figure 5 and Eq. (2) demonstrate that the sum of two "one-dimensional (1D)" Gaussian functions equal the two-dimensional (2D) Gaussian function:

$$G(x.y) = \frac{1}{2\pi\sigma^2}\, e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (2)$$

where, σ is the standard deviation of this two-dimensional

Gaussian function, it can also be called Gaussian radius, and its corresponding value range is [0.1~250]. σ^2 is the variance. Gaussian smoothing can be used in the preprocessing step of a computer vision algorithm to improve photos of various sizes. The method of Gaussian smoothing of a picture combines the image and normal distributions. In general, Gaussian picture sliding technology is utilized at low frequencies, while high frequencies are filtered. Gaussian smoothing's function is to reduce noise while also achieving the smoothing effect [27].

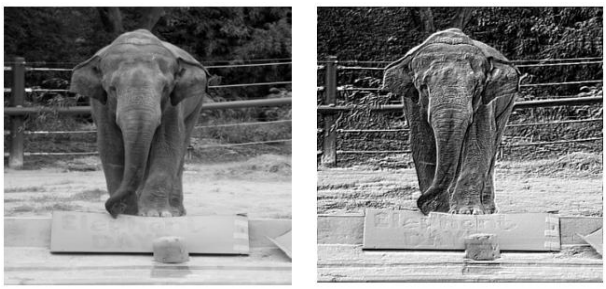

Figure 5. Blur image using Gaussian blur

3.3.3 Apply histogram equalization

Since the cumulative histogram equalization method performs well in histogram equalization, it is used when digital images have low contrast values, such as non-formal image brightening allocation or poor illumination. This way, after the input-colored images are converted to greyscale, the contrast of the greyscale image is improved [28]. The histogram was created by utilizing Eq. (3) and the result shown in Figure 6 as follows:

$$h[i]=\sum_{x=1}^{N}\sum_{y=1}^{M}\begin{cases}0 & \text{if } f[x.y]=i\\ 1 & \text{otherwise}\end{cases} \quad (3)$$

The cumulative distributions are then determined utilizing Eqs. (4) and (5) as follows:

$$cdf(X_i)=\sum_{i=0}^{k} p(X_i) \quad (4)$$

$$g[x.y]=\frac{CDF[f[x.y]-CDF_{min}]}{(N\times M)-CDF_{min}}\times(L-1) \quad (5)$$

where, (x, y) is a coordinated pixel value, and N, M is the height and width of an input image [29].

Figure 6. Histogram equalization process

3.3.4 Image resize

The size of the images used in the database is different, so it requires changing all the static and dynamic images in the databases used in this work to a fixed size for all images. In this study, Figure 7 shows the resizing 20×20 which is the best among several experiments conducted, when the scale of the image was lower the results and the features obtained were better. The idea of the suggested method depends on detecting more than one object in the image, this size was the best for extracting the most relevant features. Resizing in object detection would work much better when there are small images to a much larger size since it puts the model at regular inputs for processing. Smaller images have less information than larger ones and require less computational power. Resizing conserves feature recognition across different scales. Besides, CNNs are robust against losing some information after resizing. Proper resizing further allows the network to focus on important features while avoiding noisy overfitting and extra information. However, these results depend on the expectation that the important object features are maintained during resizing since our goal depends on computer vision and not image processing [30].

Figure 7. Image resizes by 20×20

3.4 Feature extraction phase

Feature extraction is a key technique in machine learning and signal processing that involves converting raw data into a more acceptable format for analysis or input to algorithms [31]. In fields such as computer vision, natural language processing, and signal processing, raw data often contains vast amounts of information, much of which may be irrelevant or redundant for specific tasks. The goal of feature extraction is to capture the most relevant characteristics or patterns while discarding noise [32].

Applying PCA and LBPH allows the model to focus on the most discriminative elements, thereby improving efficiency and accuracy. This synergy allows the object detection model to retain significant texture information in a reduced feature space, thus enabling more effective and faster object detection.

Most real-world data require multiple features to be used, for instance, extracted through PCA and LBPH, due to inherent complexity and variability. Different feature extraction methods usually grasp different aspects of the data.

- Diverse Information: The model will be able to take advantage of both the global structural information from PCA and the local texture information provided by LBPH. The dual perspective allows for better discrimination in the feature set, thus enabling more robust recognition and classification.
- Improved Robustness: Integration of features derived from different methodologies will enhance the robustness of the model against variations of the input data, be it changes in lighting, pose, or even occlusion. This becomes more important in applications related to face recognition due to these varieties.
- Better Performance: Several empirical studies show that most models developed based on feature combinations

tend to outperform those based on one feature extraction method. This is explained by the fact that capturing a wide range of characteristics from the data enables the model to achieve better accuracy and generalization.

3.4.1 Feature extraction depending on (PCA)

Principle Component Analysis (PCA) is a statistical method that employs a comprehensive approach to detecting patterns in high-dimensional data. It derives from the information theory method, which divides pictures into tiny groups of distinctive feature images known as Eigens. This method is crucial in recognition technology for identifying and verifying features. The 2-dimensional image matrices are transformed into a 1-dimensional vector, which can be either a row or column vector [33]. The steps of PCA are as follows:

Raw data standardization: The raw data should have a unit variance and a zero mean.

$$X_j^i = \frac{x_j^i - \bar{x}_j}{\sigma_j} \forall j \tag{6}$$

Compute the raw data's covariance matrix as follows:

$$\sum = \frac{1}{m}\sum_i^m (X_i)\ (X_i)^T, \sum \in R^{n*n} \tag{7}$$

Compute the covariance matrix's eigenvector and eigenvalue as presented in Eq. (8).

$$u^T \sum = \mu\lambda \tag{8}$$

$$U = \begin{bmatrix} | & | & | \\ u_1 & u_2 & u_3 \\ | & | & | \end{bmatrix}, u_i \in R^n \tag{9}$$

where, $\frac{1}{m}$ in this model equaled $\frac{1}{20}$ cause PCA here is used as feature extraction, to work better, the image dimensions should be equal.

The raw data must be transformed into a k-dimensional space as follows: The covariance matrix's top k eigenvectors are chosen. These will be the new original foundation for the data. The corresponding vector is calculated using Eq. (10).

$$x_i^{new} = \begin{bmatrix} u_1^T & x^i \\ u_2^T & x^i \\ \cdots & \\ u_i^T & x^i \end{bmatrix} \in R^k \tag{10}$$

If the original data has n dimensions, it will be transformed into a new k-dimensional representation [34].

3.4.2 Feature extraction depending on LBPH

The original LBP operator is a strong tool for texture description [35]. The operator labels an image's pixels by thresholding the 3×3-neighborhood of each pixel with the center value and viewing the result as a binary integer, as illustrated in Eq. (11) illustrating the basic LBP operator [36].

$$LBP(x_c, y_c) = \sum_{P=0}^{7} S(i_p - i_c)2^P \tag{11}$$

where, i_p and i_c Are neighbors and central pixel values respectively,

$$S(t) = \begin{cases} 1 & t \geq 0 \\ 0 & t < 0 \end{cases}$$

For neighbors equal to eight for each label, there are 256 possible combinations (28=256).

3.5 Detection phase based on proposed object-CNN

The proposed object-CNN architecture presents a sophisticated and innovative approach to object detection and recognition. It leverages a carefully crafted combination of neural network layers designed to achieve superior performance. At its core, the architecture integrates 1D CNNs, a cornerstone of modern computer vision, with additional layers specifically tailored for object detection tasks [37].

These layers include:

- Convolutional Layers: Responsible for feature extraction from the input image.
- Pooling Layers: Used to down-sample and reduce dimensionality, enhancing computational efficiency.
- Fully Connected Layers: Act as classifiers to interpret the extracted features.
- Dense layers: connect every neuron from the previous layer to every neuron in the current layer. They serve to aggregate and combine the features extracted from convolutional layers, allowing the model to learn complex relationships and make final predictions.
- LeakyReLU: applied to the convolutional layers or dense layers to enhance the model's ability to learn and represent the object detection task more effectively.
- Flatten: it converts the multi-dimensional output from convolutional operations into a flat 1D array, making it possible to be processed by subsequent Dense layers.
- Outputting bounding box coordinates through regression, using anchor boxes, and applying NMS to finalize the detected bounding boxes. [38].

What sets the proposed object-CNN apart is the meticulous placement and fine-tuning of these layers, allowing the model to effectively utilize information in both space and time across consecutive frames in video sequences. This design enables accurate object detection and recognition over time. The object-CNN boasts remarkable detection accuracy while ensuring computational efficiency, making it a powerful tool for a variety of real-world applications. One notable advantage of 1D CNNs [39] is their low computational requirements, making them suitable for real-time hardware implementation. Their simple and compact configuration allows the 1D object-CNN to perform only one-dimensional convolution operations efficiently.

The detailed architecture of the proposed 1D object-CNN model, which consists of twenty-eight layers, is presented in Table 2:

- Nine convolution layers (Conv1D) with hyperparameter:
  - filters (16, 32, 64, 64, 32,32,16,16 and 485)
  - kernel size=3
  - stride=1
  - padding=valid
- Seven layers of Max-pooling 1D with
  - pool size=1
  - strides=1
- Eight Leaky-ReLU with
  - lpha=0.3

- Three fully connected layers are represented by (Dense)
  - Dense1 (128, activation=linear)
  - Dense2 (512, activation=linear)
  - Dense3 (92, activation=softmax)
- One flattened layer.
- optimizers.Adam(lr=0.001)
- epoch=100

Table 2. Summary of components for the proposed model's architecture

Layer No.	Layer Type	Output Shape	#Param
1	Conv-1	(None, 398, 16)	64
	MaxPooling-1	(None, 398, 16)	0
	LeakyReLU-1	(None, 398, 16)	0
2	Conv-2	(None, 396, 32)	1568
	MaxPooling-2	(None, 396, 32)	0
	LeakyReLU-2	(None, 396, 32)	0
3	Conv-3	(None, 394, 64)	6208
	MaxPooling-3	(None, 394, 64)	0
	LeakyReLU-3	(None, 394, 64)	0
4	Conv-4	(None, 392, 64)	12352
	MaxPooling-4	(None, 392, 64)	0
	LeakyReLU-4	(None, 392, 64)	0
	Dense-1	(None, 392, 128)	8320
5	Conv-5	(None, 390, 32)	12320
	MaxPooling-5	(None, 390, 32)	0
	LeakyReLU-5	(None, 390, 32)	0
6	Conv-6	(None, 388, 32)	3104
	MaxPooling-6	(None, 388, 32)	0
	LeakyReLU-6	(None, 388, 32)	0
	Dense-2	(None, 388, 512)	16896
7	Conv-7	(None, 388, 16)	24592
	LeakyReLU-7	(None, 388, 16)	0
8	Conv-8	(None, 388, 16)	784
	LeakyReLU-8	(None, 388, 16)	0
9	Conv-9	(None, 388, 485)	23765
10	Flatten-1	(None, 188180)	11872
11	Dense-3	(None, 92)	17312652
Total Param:		**17,422,625**	
Trainable Param:		**17,422,625**	
Non-trainable Param:		**0**	

The proposed model is made up of nine convolutional layers, each followed by one pooling layer. The numbers of filters used with the convolution layer are (16, 32, 64, 64, 32, 32,16, 16 and 485) respectively. These filters are applied stride of these kernels used is one for each window. The usage of CNN as a non-linear layer, most often with "leaky Rectified Linear Units" (LeakyReLU) [40] is an activation function utilized for all levels of the model and a sigmoid [41] for the last dense layer," can be used to build a variety of functions (output layer) [42]. This model uses the max-pooling function for the pooling layers. These CNNs are formed by allocating the input samples into non-overlapping one-dimensional areas, treating each space as a cluster, and choosing the maximum value from each space. The final layer is the fully connected layer also called the output layer or decision-making layer which contains three neurons and implements a sigmoid function to provide the final class of the sequences. The model's learning ability is improved by including tiny convolution kernels. To keep the model from overfitting, the original entire connection layer is eliminated. The activation function is Leaky ReLU. Here, is the explanation of the reason behind using these layers in this order:

1) Increasing Depth with Filters
- *Initial Layers:* The first group of layers normally contains fewer filters such as 16, and 32 because they try to learn lower-level features from the input. On the forefront of a network, the filters shall be designed to identify just the edges or other minor temporal features in the data.
- *Middle Layers:* Where most of the depth in the network is often accompanied by an increase in the number of filters, say to 64, to enable the model to capture more complicated patterns and high-level representations. Increased filter count helps scale its capacity to learn intricate features from data.

2) Bottleneck Architecture
- *Downsampling Filters:* Beyond the peak filter count of 64, the architecture downsamples by reducing the number of filters down to 32 and then 16. There are several reasons for this:
  a) Dimensionality Reduction: This reduces the number of filters, hence reducing computational complexity, but retaining the most important information. It may also serve as a form of regularization to prevent overfitting.
  b) Feature Combinations: The model combines features learned from earlier layers through training to help in the representation of data with a lesser number of filters.

3) Final Layers and Output
- *Transition to Output:* Often the last few layers have a lesser number of filters, 16, for example, that summarize the features into a representation that is manageable for the output layer. This helps transition from feature maps to class probabilities or predictions.
- *Special Case Filter (485):* this would represent the number of classes or unique outputs produced by the model. In detection, the number of filters in the final Dense layer corresponds to the number of classes.

4) Adaptive Design

The architecture we specified is adaptive to the problem at hand. Suppose there are a lot of varying features in the dataset, such as temporal and frequency; then, it would be better to have a higher number initially to capture all possible variations early in the network. These features then get refined into essential features for prediction by later layers.

5) Empirical Results

We use convolutional layers; these layers apply one-dimensional convolution to the input data, which is particularly useful for sequence data. A kernel size of (3, 1) means that each filter will extend over a length of 3 units in the input data; this allows the model to capture local patterns or features. The first dimension (None) represents the batch size and hence can vary, which means the model can handle any number of input samples. For each convolution layer, we have one MaxPooling1D Layer; hence we are using seven MaxPooling layers in all. This layer is used to reduce the dimensionality of feature maps while still maintaining the most critical information. It helps in overfitting reduction and cost computation. After each pooling layer, we used seven activation functions. The Leaky ReLU is an activation function that allows a small gradient when the input is negative. This goes into introducing nonlinearities in the model, allowing.

4. RESULTS AND DISCUSSION

This work presents a carefully designed object-CNN model

that seamlessly integrates data preprocessing, where images are optimized to enhance critical structural features while suppressing irrelevant noise. Subsequently, the hybrid feature extraction strategy-merging yields a robust 1D vector of 800 features (where PCA extracted 400 features, LBPH extracted 400 features) that effectively captures the essence of the input images.

The proposed object-CNN model, which integrates a one-dimension CNN with feature extraction techniques achieved exceptional performance on the COCO dataset, with accuracy [43], precision, recall, and F-measure [44] all reaching 99.74%, as shown in Table 3. This achievement underscores the model's effectiveness and robustness, demonstrating its ability to capture complex patterns within the dataset while generalizing well to unseen instances (with an average of 28 seconds per epoch).

Two-stage detectors, like Faster R-CNN (87.7% accuracy, 84.5% precision, trained over an unspecified number of epochs) [8] and Cascade R-CNN (88.2% accuracy, trained over an unspecified number of epochs) [9], excel in accuracy through their sophisticated region proposal mechanisms, offering good detection capabilities at the cost of slower performance unsuitable for real-time applications. In contrast, single-stage detectors such as YOLOv9 (93.5% accuracy, 91.0% precision, trained on the COCO dataset) [14] and EfficientDet-D7 (89.5% accuracy, Precision 84.0%, Recall 82.0%, trained over 300 epochs) [12] provide faster inference speeds, making them ideal for real-time scenarios but sometimes sacrificing accuracy in cluttered environments. Object-CNN must demonstrate competitive accuracy, ideally exceeding 88%, and maintain efficient inference times to position itself as a viable alternative within this competitive landscape. Additionally, the model's complexity, the number of epochs for training, and resource requirements will play crucial roles in its deployment potential, particularly in mobile and embedded systems, where lightweight architectures like MobileNetV2 (71.8% accuracy, trained over 100 epochs) [7] have found success despite lower performance metrics. A thorough empirical evaluation as in Figure 8 of Object-CNN against these established benchmarks [6, 10, 11, 15, 16] is imperative to ascertain its effectiveness and applicability in practical scenarios, ensuring it offers distinctive advantages over existing models.

The design of the loss function and the choice of optimization method are critical in shaping a deep learning model's learning efficiency and generalization capability. The loss function quantifies the difference between predicted outputs and actual target values, with options like cross-entropy loss for classification tasks and mean squared error for regression tasks serving distinct purposes. Sensitivity analysis can evaluate the impact of different loss functions, identifying the most suitable for specific applications and potentially leading to improved performance, especially in imbalanced datasets. Similarly, optimization methods such as Stochastic Gradient Descent (SGD), Adam, and RMSProp influence how model weights are updated based on gradients, affecting convergence speed and stability. Conducting sensitivity analysis on optimization strategies reveals their effects on training dynamics, where adaptive learning rate methods like Adam often yield faster convergence and better performance. The interplay between loss function design and optimization method is paramount, as their combined effect determines the model's ability to avoid overfitting and underfitting, with tailored loss functions including regularization terms further mitigating such risks. Ultimately, evaluating performance through metrics like accuracy, precision, recall, and F1-score reflects the effectiveness of these strategies, illustrating how robust models that leverage thoughtful loss and optimization designs can excel across various tasks.

The suggested model's accuracy, as shown in Figure 9, demonstrates high initial values of 99.83% for training and 99.74% for testing, reflecting a steady improvement over several epochs. The training accuracy consistently rises, indicating the model's growing skill in predicting training data, while validation accuracy closely matches, suggesting effective generalization to unseen data.

In terms of loss, Figure 10 reveals a decrease from an initial training loss of 0.1013 and a validation loss of 0.1647, indicating the model is learning effectively. The consistent reduction in both training and validation losses, alongside increasing accuracy, signifies a well-balanced model performance without signs of overfitting or underfitting [45]. This progressive improvement throughout the epochs underscores the model's capability to optimize parameters, reduce errors, and enhance predictive accuracy, affirming its robustness in both training and validation datasets.

Table 3. Experimental results of the proposed system

Model	Accuracy	Precision	Recall	F-Measure	Time Per Epoch
Object-CNN-PCA-LBPH	99.74%	99.74%	99.74%	99.74%	28 sec.

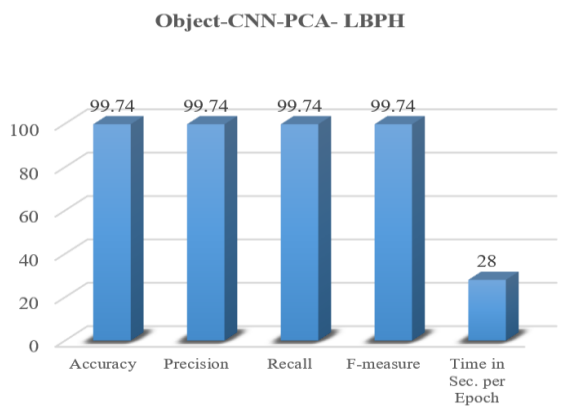

Figure 8. Chart of evaluation results of the proposed model

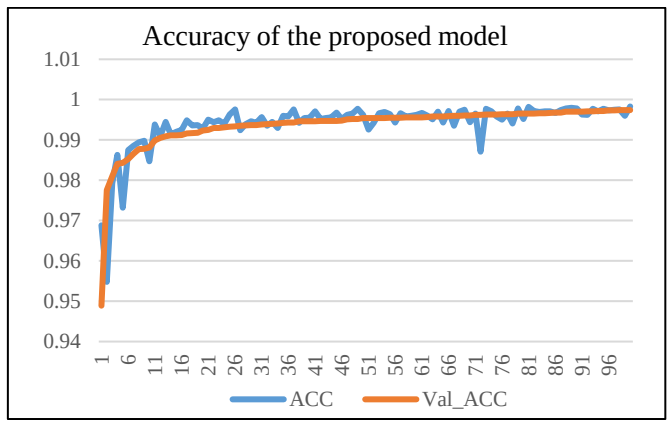

Figure 9. Accuracy results of the proposed model

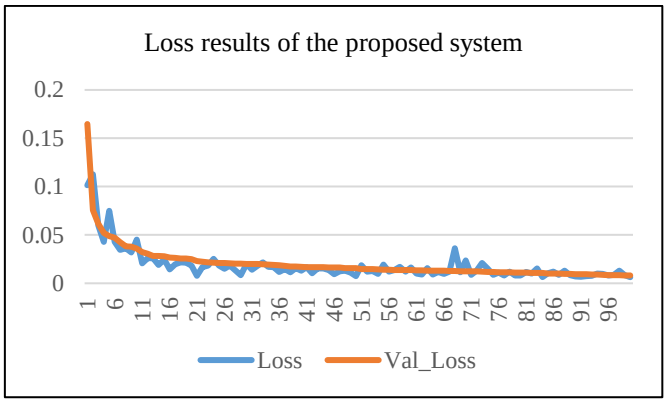

Figure 10. Loss results of the proposed model

5. CONCLUSION

In this paper, we demonstrated and evaluated a deep learning model designed to optimize accuracy and minimize loss. The model achieved impressive accuracy rates of 99.83% in training and 99.74% in testing, demonstrating effective learning and robust generalization capabilities. Our analysis revealed a consistent upward trend in accuracy and a simultaneous decrease in both training and testing losses, indicative of the model's ability to optimize its parameters over several epochs without exhibiting signs of overfitting or underfitting. These results imply that the model not only fits the training data well but also performs well on unknown data, indicating its potential for use in real-world applications.

However, some limitations could be addressed in future work, including the model's complexity, which restricts it to scenarios involving at least two classes, as the presence of multiple classes is a fundamental requirement. Additionally, while the model is capable of detecting and recognizing objects, it does not extend to classifying them.

Overall, the findings affirm that thoughtful design choices and continuous evaluation throughout the training process are essential to developing deep learning models that achieve high accuracy and effectively address complex tasks in various domains. Future work will focus on further refining the model's architecture, exploring additional optimization techniques, and enhancing its functionality to improve performance and applicability in broader contexts.

REFERENCES

[1] Rahaman, M. (2023). The current trends of object detection algorithms: A review. https://doi.org/10.13140/RG.2.2.19067.49442

[2] Deng, Z., Li, A. (2024). Object detection algorithms based on convolutional neural networks. Highlights in Science. Engineering and Technology, 81: 243-251. https://doi.org/10.54097/vyfg4e34

[3] Saleh, K., Szénási, S., Vámossy, Z. (2021). Occlusion handling in generic object detection: A review. In 2021 IEEE 19th World Symposium on Applied Machine Intelligence and Informatics (SAMI), Herl'any, Slovakia, pp. 000477-000484. https://doi.org/10.1109/SAMI50585.2021.9378657

[4] Srivastava, S., Divekar, A.V., Anilkumar, C., Naik, I., Kulkarni, V., Pattabiraman, V. (2021). Comparative analysis of deep learning image detection algorithms. Journal of Big Data, 8(1): 66. https://doi.org/10.1186/s40537-021-00434-w

[5] Pardeshi, S., Wagh, N., Kharat, K., Pawar, V., Yannawar, P. (2023). A novel approach for object detection using optimized convolutional neural network to assist visually impaired people. In First International Conference on Advances in Computer Vision and Artificial Intelligence Technologies (ACVAIT 2022). Atlantis Press, pp. 187-207. https://doi.org/10.2991/978-94-6463-196-8_17

[6] He, K., Zhang, X., Ren, S., Sun, J. (2019). Identity mappings in deep residual networks. arXiv preprint arXiv:1603.05027. https://doi.org/10.48550/arXiv.1603.05027

[7] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4510-4520. https://doi.org/10.48550/arXiv.1801.04381

[8] Zhang, W., Zhu, Q., Li, Y., Li, H. (2023). MAM faster R-CNN: Improved faster R-CNN based on malformed attention module for object detection on X-ray security inspection. Digital Signal Processing, 139: 104072. https://doi.org/10.1016/j.dsp.2023.104072

[9] Moosmann, J., Giordano, M., Vogt, C., Magno, M. (2023). TinyissimoYOLO: A quantized, low-memory footprint, tinyml object detection network for low power microcontrollers. In 2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS), Hangzhou, China, pp. 1-5. https://doi.org/10.1109/AICAS57966.2023.10168657

[10] Krichen, M. (2023). Convolutional neural networks: A survey. Computers, 12(8): 151. https://doi.org/10.3390/computers12080151

[11] Alsuwaylimi, A.A., Alanazi, R., Alanazi, S.M., Alenezi, S.M., Saidani, T., Ghodhbani, R. (2024). Improved and efficient object detection algorithm based on yolov5. Engineering, Technology & Applied Science Research, 14(3): 14380-14386. https://doi.org/10.48084/etasr.7386

[12] Tan, M., Pang, R., Le, Q.V. (2020). Efficientdet: Scalable and efficient object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10781-10790. https://doi.org/10.48550/arXiv.1911.09070

[13] Afdhal, A., Saddami, K., Sugiarto, S., Fuadi, Z., Nasaruddin, N. (2023). Real-Time object detection performance of yolov8 models for self-driving cars in a mixed traffic environment. In 2023 2nd International Conference on Computer System, Information Technology, and Electrical Engineering (COSITE), Banda Aceh, Indonesia, pp. 260-265. https://doi.org/10.1109/COSITE60233.2023.1024952

[14] Santos Júnior, E.S.D., Paixão, T., Alvarez, A.B. (2025). Comparative performance of YOLOv8, YOLOv9, YOLOv10, and YOLOv11 for layout analysis of historical documents images. Applied Sciences, 15(6): 3164. https://doi.org/10.3390/app15063164

[15] Sapkota, R., Meng, Z., Churuvija, M., Du, X., Ma, Z., Karkee, M. (2024). Comprehensive performance evaluation of yolo11, yolov10, yolov9, and yolov8 on detecting and counting fruitless in complex orchard environments. arXiv Preprint arXiv: 2407.12040. https://doi.org/10.48550/arXiv.2407.12040

[16] Pujara, A., Bhamare, M. (2022). DeepSORT: Real time

& multi-object detection and tracking with YOLO and TensorFlow. In 2022 International Conference on augmented intelligence and sustainable systems (ICAISS), Trichy, India, pp. 456-460. https://doi.org/10.1109/ICAISS55157.2022.10011018

[17] Adarsh, P., Rathi, P., Kumar, M. (2020). YOLO v3-Tiny: Object detection and recognition using one stage improved model. In 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, pp. 687-694. https://doi.org/10.1109/ICACCS48705.2020.9074315

[18] ghani Abdulghani, A.M.A., Dalveren, G.G.M. (2022). Moving object detection in video with algorithms YOLO and faster R-CNN in different conditions. Avrupa Bilim ve Teknoloji Dergisi, (33): 40-54. https://doi.org/10.31590/ejosat.1013049

[19] Diwan, T., Anirudh, G., Tembhurne, J.V. (2023). Object detection using YOLO: Challenges, architectural successors, datasets and applications. Multimedia Tools and Applications, 82(6): 9243-9275. https://doi.org/10.1007/s11042-022-13644-y

[20] Yadav, S.P., Jindal, M., Rani, P., de Albuquerque, V.H.C., dos Santos Nascimento, C., Kumar, M. (2024). An improved deep learning-based optimal object detection system from images. Multimedia Tools and Applications, 83(10): 30045-30072. https://doi.org/10.1007/s11042-023-16736-5

[21] Xu, Y., Goodacre, R. (2018). On splitting training and validation set: A comparative study of cross-validation, bootstrap and systematic sampling for estimating the generalization performance of supervised learning. Journal of Analysis and Testing, 2(3): 249-262. https://doi.org/10.1007/s41664-018-0068-2

[22] Rácz, A., Bajusz, D., Héberger, K. (2021). Effect of dataset size and train/test split ratios in QSAR/QSPR multiclass classification. Molecules, 26(4): 1111. https://doi.org/10.3390/molecules26041111

[23] Fan, C., Chen, M., Wang, X., Wang, J., Huang, B. (2021). A review on data preprocessing techniques toward efficient and reliable knowledge discovery from building operational data. Frontiers in Energy Research, 9: 652801. https://doi.org/10.3389/fenrg.2021.652801

[24] Strasser, S., Klettke, M. (2024). Transparent data preprocessing for machine learning. In Proceedings of the 2024 Workshop on Human-In-the-Loop Data Analytics, pp. 1-6. https://doi.org/10.1145/3665939.3665960

[25] Baek, H.S., Kim, J., Jeong, C., Lee, J., Ha, J., Jo, K., Kim, M.H., Sohn, T.S., Lee, I.S., Lee, J.M., Lim, D.J. (2024). Deep learning analysis with gray scale and doppler ultrasonography images to differentiate graves' disease. The Journal of Clinical Endocrinology & Metabolism, 109(11): 2872-2881. https://doi.org/10.1210/clinem/dgae254

[26] Rathore, Y.K., Janghel, R.R., Swarup, C., Pandey, S.K., Kumar, A., Singh, K.U., Singh, T. (2023). Detection of rice plant disease from RGB and grayscale images using an LW17 deep learning model. Electronic Research Archive, 31(5): 2813-2833. https://doi.org/10.3934/era.2023142

[27] Xi, E., Li, M. (2022). Research on the enhancement algorithm of defocused and blurred image base on non-Local constraints. International Journal of Circuits, Systems and Signal Processing, 16: 934-940. https://doi.org/10.46300/9106.2022.16.114

[28] Mungra, D., Agrawal, A., Sharma, P., Tanwar, S., Obaidat, M.S. (2020). PRATIT: A CNN-based emotion recognition system using histogram equalization and data augmentation. Multimedia Tools and Applications, 79(3): 2285-2307. https://doi.org/10.1007/s11042-019-08397-0

[29] Chowdhury, J.H., Liu, Q., Ramanna, S. (2024). Simple histogram equalization technique improves performance of VGG models on facial emotion recognition datasets. Algorithms, 17(6): 238. https://doi.org/10.3390/a17060238

[30] Kumar, D., Rajaan, R., Choudhary, D., Sharma, D. A comprehensive review and comparison of image super-resolution techniques. International Journal of Advanced Engineering, Management and Science, 10: 40-45. https://doi.org/10.22161/ijaems.102.5

[31] Escobar-Linero, E., Luna-Perejon, F., Munoz-Saavedra, L., Sevillano, J.L., Domínguez-Morales, M. (2022). On the feature extraction process in machine learning. An experimental study about guided versus non-Guided process in falling detection systems. Engineering Applications of Artificial Intelligence, 114: 105170. https://doi.org/10.1016/j.engappai.2022.105170

[32] Jayalaxmi, P.L.S., Saha, R., Kumar, G., Kim, T.H. (2022). Machine and deep learning amalgamation for feature extraction in Industrial Internet-of-Things. Computers & Electrical Engineering, 97: 107610. https://doi.org/10.1016/j.compeleceng.2021.107610

[33] Choi, S.W., Kim, B.H. (2021). Applying PCA to deep learning forecasting models for predicting PM2.5. Sustainability, 13(7): 3726. https://doi.org/10.3390/su13073726

[34] Wang, D., Su, J., Yu, H. (2020). Feature extraction and analysis of natural language processing for deep learning English language. IEEE Access, 8: 46335-46345. https://doi.org/10.1109/ACCESS.2020.2974101

[35] Mubarak, A.S., Serte, S., Al-Turjman, F., Ameen, Z.S.I., Ozsoz, M. (2022). Local binary pattern and deep learning feature extraction fusion for COVID-19 detection on computed tomography images. Expert Systems, 39(3): e12842. https://doi.org/10.1111/exsy.12842

[36] Deeba, F., Memon, H., Dharejo, F.A., Ahmed, A., Ghaffar, A. (2019). LBPH-based enhanced real-time face recognition. International Journal of Advanced Computer Science and Applications, 10(5). https://doi.org/10.14569/IJACSA.2019.0100535

[37] Adjabi, I., Ouahabi, A., Benzaoui, A., Taleb-Ahmed, A. (2020). Past, present, and future of face recognition: A review. Electronics, 9(8): 1188. https://doi.org/10.3390/electronics9081188

[38] Purwono, P., Ma'arif, A., Rahmaniar, W., Fathurrahman, H.I.K., Frisky, A.Z.K., ul Haq, Q.M. (2022). Understanding of convolutional neural network (CNN): A review. International Journal of Robotics and Control Systems, 2(4): 739-748. https://doi.org/10.31763/ijrcs.v2i4.888

[39] Zhao, L., Zhang, Z. (2024). A improved pooling method for convolutional neural networks. Scientific Reports, 14(1): 1589. https://doi.org/10.1038/s41598-024-51258-6

[40] Goodwin, M., Halvorsen, K.T., Jiao, L., Knausgård, K. M., Martin, A.H., Moyano, M., Oomen, R.A., Rasmussen, J.H., Sørdalen, T.K., Thorbjørnsen, S.H. (2022). Unlocking the potential of deep learning for

marine ecology: Overview, applications, and outlook. ICES Journal of Marine Science, 79(2): 319-336. https://doi.org/10.1093/icesjms/fsab255

[41] Khan, H., Haq, I.U., Munsif, M., Mustaqeem, Khan, S.U., Lee, M.Y. (2022). Automated wheat diseases classification framework using advanced machine learning technique. Agriculture, 12(8): 1226. https://doi.org/10.3390/agriculture12081226

[42] Hicks, S.A., Strümke, I., Thambawita, V., Hammou, M., Riegler, M.A., Halvorsen, P., Parasa, S. (2022). On evaluation metrics for medical applications of artificial intelligence. Scientific Reports, 12(1): 5979. https://doi.org/10.1038/s41598-022-09954-8

[43] Joseph, V.R., Vakayil, A. (2022). SPlit: An optimal method for data splitting. Technometrics, 64(2): 166-176. https://doi.org/10.1080/00401706.2021.1921037

[44] Arumugam, S.R., Gowr, S., Abimala, Balakrishna, Manoj, O. (2022). Performance evaluation of machine learning and deep learning techniques: A comparative analysis for house price prediction. Convergence of Deep Learning in Cyber-IoT Systems and Security, 21-65. https://doi.org/10.1002/9781119857686.ch2

[45] Rainio, O., Teuho, J., Klén, R. (2024). Evaluation metrics and statistical tests for machine learning. Scientific Reports, 14(1): 6086. https://doi.org/10.1038/s41598-024-56706-x