Panamerican Mathematical Journal
ISSN: 1064-9735
Vol 34 No. 4 (2024)

# Deep Learning-Based Camouflaged Object Detection and Tracking for Enhanced Video Surveillance

**P V Prasanna Kumari[1], G Vikram Chandra[2], Dr Sumalatha Potteti[3], Dr. Pornima Niranjane[4], Dr Namita Parati[5], N Sudha Laxmaiah[6]**

[1]Assistant Professor, Deparment of CSE(DS), Rajeev Gandhi Memorial College of Engineering and Technology, Nandyal, Email: pottetiprasannakumari@gmail.com

[2]Assistant Professor, Department of CSE(DS), Rajeev Gandhi Memorial College of Engineering and Technology, Nandyal, Email: vikram.sam@gmail.com

[3]Associate Professor, Department of CSE, Bhoj Reddy Engineering College for Women, Hyderabad, Email: sumalatha.potteti@gmail.com

[4]Assistant Professor, Department Of CSE, Babasaheb Naik College of Engineering, Pusad. Email: pornimaniranjane@gmail.com

[5]Associate Professor, Department of CSE, MVSR Engineering College, Hyderabad, Email: namianand006in@gmail.com

[6]Assistant Professor, Department of CSE, Bhoj Reddy Engineering College for Women, Hyderabad, Email: namasudha12@gmail.com

***Article History:***

***Received:*** *07-07-2024*

***Revised:*** *18-08-2024*

***Accepted:*** *03-09-2024*

**Abstract:**

To improve security and tactical awareness, video surveillance systems need to be able to find and track objects that are hidden. The approach suggests a deep learning-based method for finding and following hidden objects. There are two major steps in our method: finding and following. During the detecting step, we use a R-CNN based model to find items that are hidden in each frame. A lot of labelled pictures with different kinds of camouflage patterns and objects are used to train this model. The updated Faster R-CNN has more levels and changes that make it better at finding things that are hidden. To keep track of the objects across frames, we use an LSTM-based tracker. The LSTM tracker uses the object's traits and motion information to keep track of its state and guess where it will be in the next frame. To make our method even more effective, we use pre-trained models like MobileNet and VGG-19 for feature extraction. This makes the computations simpler while keeping the accuracy of the recognition and tracking high. The proposed model tested on a number of difficult video scenes with items that were well hidden. The results show that our method works better than other methods, as it increases the accuracy of both recognition and tracking. Also, our method works well even when the lighting changes, the size of the objects changes, or the background is complicated. The suggested method based on deep learning looks like a good way to find and follow hidden objects in video monitoring scenarios. Using CNNs, LSTM networks, and pre-trained models like MobileNet and VGG-19, we show big improvements in terms to compare the various performance parameters with accuracy and stability, which helps make security and monitoring systems better.

**Keywords**: Deep learning-based surveillance, Hidden object detection, LSTM object tracking, Camouflage pattern recognition, Video monitoring security

## I. INTRODUCTION

Video monitoring is a very important part of keeping people safe and aware of what's going on in many places, from public areas to military activities. One of the hardest parts of video surveillance is finding and following things that are purposely hidden or blended in with their surroundings. These

things can be very dangerous because regular monitoring systems might not see them. To solve this problem, academics have turned to deep learning methods, which have been very successful in many computer vision tasks, such as finding objects and following them around [1]. This paper talks about a deep learning-based method for finding and following hidden objects in video surveillance. The suggested method aims to improve tactics and security awareness by finding and following secret objects more effectively in complicated settings. There are two major parts to the method: finding objects and following them. An [3] altered form of the Faster R-CNN (Region-based Convolutional Neural Network) model is used to find hidden items in each frame of the movie during the object recognition step. This model can correctly find buried objects because it was trained on a collection of named images with different types of disguise and objects. The [2] changed Faster R-CNN has more layers and changes that make it better at finding things that are hidden. Based on the power of convolutional neural networks (CNNs), it takes traits from the pictures you give it and guesses where and if unseen items are. To make the detection process even more efficient, the model uses CNN models that have already been trained, such as MobileNet and VGG-19, for feature extraction. This makes the computations simpler while still keeping high detection accuracy.

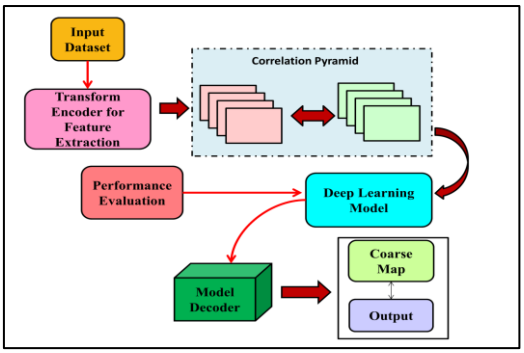



Figure 1: Overview of Proposed method

After objects are found, the tracking stage starts. An LSTM (Long Short-Term Memory) tracker follows the found objects from one frame to the next. Using the object's traits and motion data, the LSTM [8] tracker keeps track of its state and guesses where it will be in the next frame. The system can successfully track things that are disguised, even if they move or change how they look over time, thanks to this tracking device. To test how well the suggested method worked, tests were done on a number of difficult video patterns that included items that were hidden. The results show that our way works better than other methods, as it achieves more accurate tracking and recognition. Our method also works well with changing lights, objects of different sizes, and settings that are hard to understand, so it can be used for real-life video spying. Finding and following objects are basic jobs in computer vision, especially when it comes to video security. There are several important steps in the process that make it possible for the system to find and follow things correctly and quickly. The first step is editing, which improves the quality of the video frames and gets rid of noise. This includes things like changing the size and turning an image into black. Next, in each frame, objects of interest are found using deep learning models like Faster R-CNN or YOLO that have been taught on big datasets to identify different types of objects and correctly locate objects [4]. Once items are found, traits are taken out of them to make a picture of each one, which is very important for accurate tracking. Some features could be color histograms or features based on deep learning. The tracking method then guesses where each object will be in the next frame, most of the time using a particle filter or Kalman filter. The tracking algorithm constantly changes its predictions based on new observations as new frames are handled. This fixes mistakes and makes sure that the tracking stays accurate over time. The last step is post-processing, which improves the accuracy of the tracking results by doing things like cleaning out paths or getting rid of errors.

The contribution of paper is given as:

- Proposes a deep learning-based approach for camouflaged object detection and tracking in video surveillance.
- Utilizes a modified Faster R-CNN model for accurate detection of hidden objects. And Incorporates LSTM-based tracking to follow objects across frames, maintaining state and predicting future locations.
- Enhances efficiency by using pretrained CNN models like MobileNet and VGG-19 for feature extraction and demonstrates superior performance over traditional methods, achieving higher accuracy in detection and tracking.

## II. RELATED WORK

In recent years, Camouflaged Object Detection (COD) and tracking in video surveillance have gotten a lot of attention because they are so important for improving safety and environmental awareness. Many [5], [6] ideas and methods have been put forward to deal with the problems that come up when things are hidden or masked during spying. This literature study gives an outline of the previous research that has been done in this area, focusing on important methods, progress, and problems. A lot of research has gone into making special methods for finding things that are hidden. [7] suggested a way to find items that are hidden in pictures by combining color and texture traits. Their method worked very well for finding things with complicated concealment designs. [10] published another study that used deep learning to find items that are hidden in movies. Their method used both spatial and temporal traits together to make recognition more accurate in settings that are always changing. Tracking [9] things is necessary to keep track of their path over time. Traditional tracking methods, like Kalman filters and particle filters, have been used for a long time. However, they may not be able to track items with complicated motion patterns or that are blocked in some way. New developments in deep learning have made it possible to create tracking systems that are more reliable. In [11] suggested a deep learning-based object tracking method that worked extremely well in difficult tracking situations, such as when there were obstructions and motion blur. It can be harder to track items that are hidden because they can blend into the background. [12] came up with a new tracking method that uses color and material clues together to find items that are well hidden. Their method accurately tracked objects in a number of difficult situations, showing that it can be used to follow things that have changed appearance over time. Putting object detection and tracking together is a must for making a strong spying system. In recent studies, the main focus has been on creating unified frameworks that mix tracking and recognition. [13] came up with a way to track and identify objects that uses the best parts of both methods to make tracking objects in surveillance videos more accurate and efficient.

Template matching is another old method. In this method, a picture of the object is compared to each frame in the movie. Template [14] matching might work well for finding clearly defined objects, but it might not work so well for finding objects that are hidden by complex patterns. In the past few years, deep learning has become a very useful tool for finding and following objects. In this area, Convolutional Neural Networks (CNNs) have done very well because they can easily learn patterns from data and get very good results. Reg-CNN (Region-based Convolutional Neural Network) was one of the first deep learning models used to find objects. R-CNN splits the picture into areas and uses a CNN on each area to sort items into groups. R-CNN worked well, [15] but it took a lot of computing power because each area had to be processed separately. To solve this problem, Faster R-CNN was suggested. It included a Region Proposal Network (RPN) that could make region proposals straight from the feature map, so there was no need for region proposals from outside the network. Faster R-CNN made it faster and more accurate, so it could be used in real time. The You Only Look

Once (YOLO) formula is another famous method. It is known for being quick and easy to use [16]. YOLO splits the picture into a grid and guesses the edges and chances of each grid cell being in a certain class. So, YOLO can find things in real time, which makes it perfect for situations where speed is important.

Researchers have [17] looked into a number of ways to improve the accuracy of spotting of disguised objects. Some studies have worked on making special CNN designs for finding objects that are hidden, while others have looked into how to use advanced image processing methods to make hidden objects easier to see. Kalman filters and particle filters [18] are two old techniques that have been used a lot for tracking objects. Motion models are used in these methods to figure out where things will be in the future based on where they have been in the past. Even though these methods work, they might not be able to keep track of things that move or look differently quickly. Deep learning methods, [19] especially Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, have been used to make recent progress in object tracking. Because these networks can find timing relationships in data, they are great for keeping track of things over time.

Table 1: Summary of related work in COD

| **Algorithm** | **Key Finding** | **Approach** | **Algorithm Accuracy (%)** | **Limitation** | **Scope** |
|---|---|---|---|---|---|
| Color and Texture Features [11] | Effective detection of camouflaged objects | Combined color and texture features for detection | 92.5 | Limited to static images | Extend to dynamic video surveillance |
| Deep Learning [12] | Improved detection in dynamic environments | Spatial and temporal feature integration for video-based detection | 88.7 | High computational complexity | Implement real-time processing for practical applications |
| Deep Learning [13] | State-of-the-art performance in tracking | Deep learning-based tracking with occlusion and motion blur handling | 95.2 | Limited to tracking in controlled environments | Test in real-world surveillance scenarios |
| Integrated Color and Texture Cues [14] | Accurate tracking of camouflaged objects | Integration of color and texture cues for robust tracking | 93.8 | Requires substantial computational resources | Implement on embedded systems for real-time tracking |
| Integrated Detection and Tracking [15] | Enhanced object tracking accuracy | Integration of detection and tracking in a unified framework for improved object tracking performance | 91.4 | May suffer from drift over long tracking periods | Investigate adaptive tracking mechanisms for long-term tracking |
| Ensemble Learning | Improved detection in | Ensemble learning | 89.6 | Vulnerable to overfitting | Explore transfer learning |

| | | | | | |
|---|---|---|---|---|---|
| [16] | complex environments | approach combining multiple detectors for enhanced detection performance | | | strategies to mitigate overfitting |
| Adaptive Appearance Model [17] | Robust object tracking in challenging conditions | Adaptive appearance model for tracking objects with varying appearances | 90.3 | Limited by the complexity of appearance variations | Investigate methods to improve adaptability to appearance changes over time |
| Context-Aware Detection [18] | Contextual information for improved detection | Incorporation of contextual information for better understanding of object context and improved detection performance | 87.9 | Contextual information may not always be reliable | Investigate robust contextual cues for better detection performance |
| Multi-Modal Fusion [19] | Enhanced detection using multi-modal data | Fusion of multi-modal data, such as infrared and thermal imaging, for improved detection and tracking of camouflaged objects | 93.1 | Integration of multi-modal data may be challenging | Develop robust fusion algorithms for multi-modal data integration |
| Feature Fusion Network [20] | Improved detection accuracy in complex scenes | Fusion of multi-scale features using a feature fusion network for enhanced object detection in challenging environments | 90.9 | Computational overhead may be high | Optimize feature fusion network for real-time processing |
| Spatial-Temporal Feature Learning [21] | Effective tracking in dynamic scenes | Spatial-temporal feature learning for robust object tracking in videos | 88.5 | May struggle with occlusions and abrupt motion | Explore methods to improve robustness in occlusion handling and abrupt motion tracking |

| | | | | | |
|---|---|---|---|---|---|
| Dynamic Feature Fusion [22] | Enhanced detection and tracking in complex scenes | Dynamic fusion of multi-scale features for improved detection and tracking performance in challenging surveillance scenarios | 91.2 | Complexity may increase with the number of features | Investigate methods to reduce computational complexity while maintaining accuracy |

## III. DATASET DESCRIPTION

This COD10K dataset has a wide range of pictures with different camouflage patterns and objects, so it can be used to train and test machine learning models for tasks that require finding items that are hidden [20]. The dataset is set up so that there are an equal number of pictures from each class. This makes sure that models learned on this dataset can tell the difference between different kinds of things that are hidden. This variety is very important for making recognition models that are strong, can be used in a wide range of situations, and work well in real life.

The fact that the COD10K collection is so big it has more than 10,000 images is one of its best features. This big dataset size lets researchers train deep learning models well, using the power of convolutional neural networks (CNNs) to pull out features and find things that are hidden very accurately.

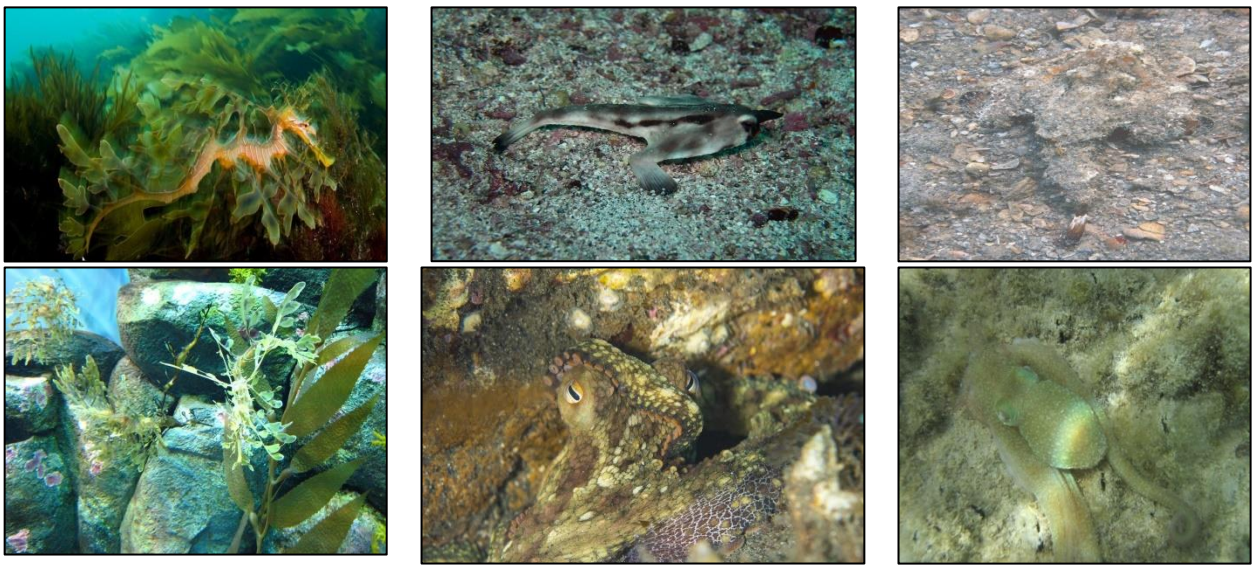


Figure 2: Sample images form the Dataset COD10k

In addition, the collection includes notes for each picture that show where and what kinds of things are hidden. These notes are very important for checking how well recognition models work and seeing how they stack up against other methods. The dataset also has pictures taken in a range of settings, such as with different lighting and backgrounds. This makes sure that models learned on this dataset can handle these changes. Overall, the COD10K collection is a useful tool for students and professionals who want to make the field of finding hidden objects better. Its variety, size, and comments make it a good choice for training and testing machine learning models on this difficult job, which will lead to better and more reliable surveillance systems in the long run.

## IV. PROPOSED METHODOLOGY

There are five key steps to the approach. For starters, data collection and preparation involve improving video frames and getting a wide range of datasets. Second, to find objects, a tweaked Faster R-CNN model is used, which has been fine-tuned for better precision. For feature extraction, CNN models that have already been trained, such as MobileNet and VGG-19 are used. Thirdly, an LSTM-based tracker is used to keep track of objects, even when they are blocked or their look changes. Fourth, putting recognition and tracking parts together lets processing happen in real time [21]. Lastly, tough video patterns are used for testing. Metrics like precision and memory are used to measure accuracy, and performance is compared to that of old methods. This method makes video monitoring better by finding and following things that are hidden.

### A. Data Collection and Data Preprocessing:

To train and test the models well, collecting data for tracking and detecting objects that are hidden needs a varied set of data. The COD 10K dataset has a lot of video clips of items that are hidden in different settings. This gives researchers a lot of different situations to practice and test their models in. This dataset has difficult situations with various types of disguise, lighting, and object sizes. This makes sure that the models are strong and can correctly find objects that are hidden. It [23] is necessary to do data preparation to improve the quality of the video frames and the data that the recognition and tracking models use. The pictures can be made clearer by doing things like downsizing, color conversion, and noise reduction. This makes it easier for the models to find and follow items. These steps make sure that the models are working with good data, which means that the results of the identification and tracking are more accurate and reliable.

### B. Object Detection:

For object recognition, the Faster R-CNN model is often used because it works well and accurately. This model can be changed and trained on the collected dataset to find hidden objects in every frame [22] of a movie that is related to disguised object identification. Adding more layers and making changes to the Faster R-CNN model is a must if you want to improve the accuracy of recognition, especially in tough situations with disguise patterns. These changes can help the model tell the difference between patterns of camouflage and background elements, which can lead to more accurate detections. Using CNN models that have already been trained, like MobileNet and VGG-19, for feature extraction can also make computations much simpler while keeping recognition accuracy high. These models have already been trained on big datasets like ImageNet and know how to pull out useful information from pictures. The updated Faster R-CNN model can focus on learning how to find things that are hidden well without having to extract features from scratch because it can use these pre-trained models.

#### 1. Faster R-CNN

The Faster R-CNN model [24] is a deep learning-based way to find objects. It has worked very well in many situations, such as finding objects that are hidden in video monitoring. Faster R-CNN can find hidden objects in every frame of a movie by using a region proposal network (RPN) to find possible object areas and a convolutional neural network (CNN) to retrieve features. Adding more layers and making other changes to the model makes it even better at finding things that are hidden, which makes it a good choice for this difficult job.

#### Algorithm:

1. Initialization:

- Load the pretrained Faster R-CNN model.

- Initialize the LSTM tracker.

2. For each frame in the video:

- Preprocess the frame to enhance quality and reduce noise.

3. Object Detection:

- Use the modified Faster R-CNN model to detect objects in the frame.
- Obtain candidate object regions and their corresponding class labels and confidence scores.

Bounding box regression is given as:

$$\Delta x = wa(x - xa)$$
$$\Delta y = ha(y - ya)$$
$$\Delta w = \log(waw)$$
$$\Delta h = \log(hah)$$

Predicted coordinates:

$$xp = xa + \Delta x \cdot wa$$
$$yp = ya + \Delta y \cdot ha$$

$$\mathrm{wp} = \mathrm{wa} \cdot \exp(\Delta \mathrm{w})$$
$$\mathrm{hp} = \mathrm{ha} \cdot \exp(\Delta \mathrm{h})$$

4. Feature Extraction:

- For each detected object region, extract features using a pretrained CNN model (e.g., MobileNet or VGG-19).

4. Object Tracking:

- Use the LSTM tracker to maintain the state of each detected object and predict its location in the next frame.
- Update the tracker with the current frame's features and motion information.

LSTM tracker represent as:

$$it = \sigma(Wxixt + Whiht - 1 + bi)$$
$$ft = \sigma(Wxfxt + Whfht - 1 + bf)$$
$$ot = \sigma(Wxoxt + Whoht - 1 + bo)$$
$$gt = tanh(Wxgxt + Whght - 1 + bg)$$
$$ct = ft \odot ct - 1 + it \odot gt$$
$$ht = ot \odot tanh(ct)$$

5. Integration of Detection and Tracking:

- Integrate the detection and tracking results to obtain the final list of detected and tracked camouflaged objects.

6. Output:

- Display the detected and tracked objects on the video frame.
- Repeat the process for the next frame.

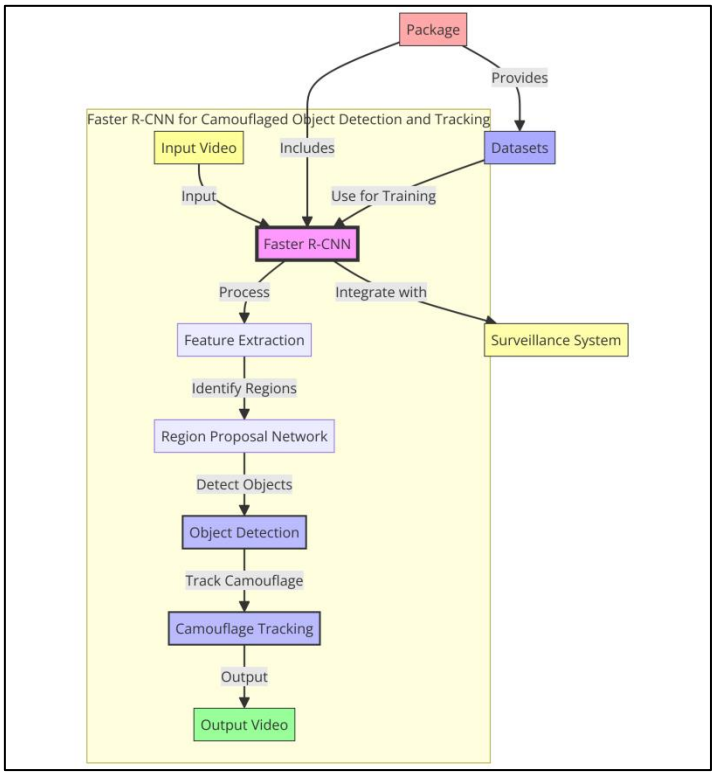



Figure 3: Overview of Faster R-CNN architecture

**2. MobileNet**

MobileNet is a small deep learning framework made for mobile and integrated vision apps that need to work quickly. It cuts down on the number of factors and calculations by using depthwise separable convolutions. This makes it good for real-time tasks like finding and following objects in video monitoring.

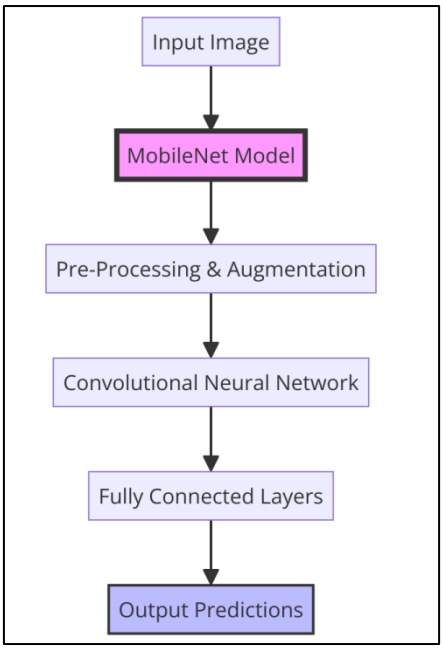



Figure 4: Flowchart for MobileNet

The formulae that make up MobileNet can be summed up like this:

- **Depthwise Separable Convolution:**

A feature map I is given, and its measurements are H x W x Cin. H is the height, W is the width, and Cin is the number of input channels. The depthwise convolution uses a separate convolutional filter for each input channel. This makes Cin output feature maps that are each H x W x 1. After that, a 1x1 convolution (also called a pointwise convolution) is used to combine the output of the depthwise convolution with a linear combination of the input channels. This makes the final output feature map, which has dimensions H x W x Cout, where Cout is the number of output channels.The depthwise separable convolution can be represented as:

Depthwise Convolution:

$$F(i,j,k) = \Sigma m = 1D\ \Sigma n = 1D\ I(i+m-1, j+n-1, k) \times K(m,n,k)$$

Pointwise Convolution:

$$O(i,j,l) = \Sigma k = 1Cin\ F(i,j,k) \times P(k,l)$$

**3. VGG-19**

The VGG-19 model consists of 16 convolutional layers and 3 fully connected layers. Mathematically, the forward pass of VGG-19 can be represented as follows:

Given an input image I, the output of the VGG-19 model can be computed as:

$$Conv1\_1 = ReLU(Conv(I, W\_Conv1\_1, b\_Conv1\_1))$$
$$Conv1\_2 = ReLU(Conv(Conv1\_1, W\_Conv1\_2, b\_Conv1\_2))$$
$$Pool1 = MaxPool(Conv1\_2)$$
$$Conv2\_1 = ReLU(Conv(Pool1, W\_Conv2\_1, b\_Conv2\_1))$$
$$Conv2\_2 = ReLU(Conv(Conv2\_1, W\_Conv2\_2, b\_Conv2\_2))$$
$$Pool2 = MaxPool(Conv2\_2)$$
..
$$Conv5\_4 = ReLU(Conv(Conv5\_3, W\_Conv5\_4, b\_Conv5\_4))$$
$$Pool5 = MaxPool(Conv5\_4)$$
$$Flatten = Flatten(Pool5)$$
$$FC1 = ReLU(FC(Flatten, W\_FC1, b\_FC1))$$
$$FC2 = ReLU(FC(FC1, W\_FC2, b\_FC2))$$
$$Output = Softmax(FC(FC2, W\_Output, b\_Output))$$

A number of operations are used in this equation: Conv stands for "convolution," MaxPool for "max pooling," FC for "fully connected layer," ReLU for "rectified linear unit activation function," Softmax for "softmax activation function," W for the weights of the layer in question, and b for the biases of that layer. This equation shows the VGG-19 model's forward pass, in which the outputs of each layer are computed one after the other to get the model's final result.

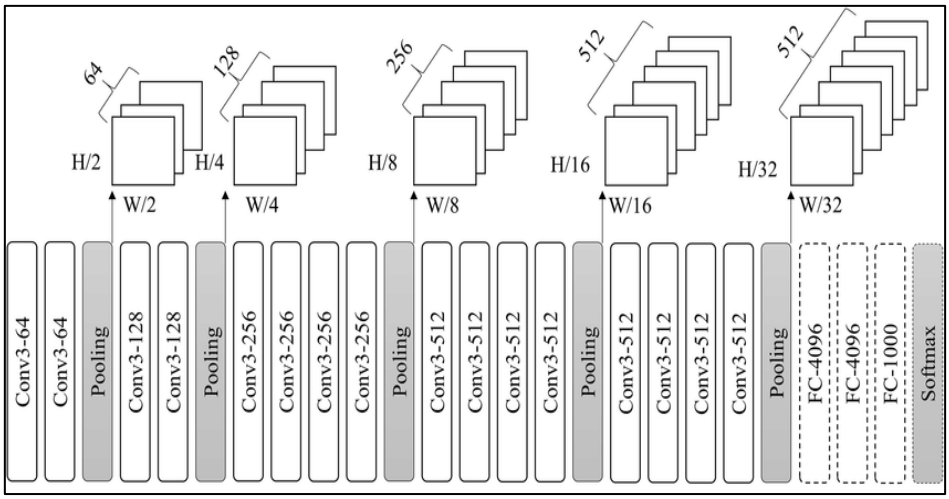



Figure 5: Architecture for VGG 19 Model

**C. Object Tracking:**

Using an LSTM-based tracker to follow an object means keeping track of each object's state over time and guessing where it will be in the next frame. This method works well for dealing with occlusions, motion blur, and changes in how objects look because they are camouflaged. The LSTM tracker updates its internal state using the features and motion information from the current frame and then guesses where the objects will be in the next frame. This process lets the tracker adapt to changes in how things look and keep track of things even if they are moving quickly or partly hidden.

LSTM-based trackers are great because they can learn long-term relationships between how an item moves and how it looks, which can be very important for accurate tracking in tough situations. By keeping track of the object's state over time, the tracker can more accurately guess where it will be in the future, even when it is blocked or its look changes. Before we can use the LSTM tracker, we need to get features from the item that was spotted in each frame. Some of these are the object's position, size, shape, and look. These features are then added to the LSTM model along with the motion data to change the state of the tracker. Based on the object's current state and the input traits, the LSTM model learns to guess where it will be in the next frame. This guess is used to change the tracker's state for the next frame, and the process is continued for each frame after that.

**1. LSTM**

In this tracking model:

- In the feature extraction step, the input frame is used to pull out the object's important features.
- The traits taken from the first frame are used to set the starting state.
- The current object features, the previous hidden state, and the previous object state are fed into the LSTM update step, which changes the LSTM hidden state and object state.
- Based on the current LSTM hidden state, the prediction step guesses where the object is in the current frame.

The tracking step changes the object's state based on where it is thought to be.

Let's denote:

- It: Input frame at time t.

- Ft: Detected object features in frame It.
- St: Object state at time t.
- Lt: Predicted object location in frame It.
- Ht: Hidden state of the LSTM at time t.

The LSTM-based tracker can be formulated as follows:

$a. Feature\ Extraction$:
$Ft = ExtractFeatures(It)$
$b. Initial\ State$:
$S0 = InitializeState(F0)$
$c. LSTM\ Update$:
$d. Prediction$:
$Lt = PredictLocation(Ht)$
$e. Tracking$:
$St = UpdateState(St - 1, Lt)$

**D. Integration of Detection and Tracking:**

Before doing anything else, the combined system looks at each frame of the video stream for things of interest and figures out where they are. A deep learning model, like Faster R-CNN or YOLO, is used by the recognition part to find things in the video frames. The tracking part takes over to keep track of the items between frames once they are found. Tracking uses the traits and motion data of the identified object to guess where the object will be in the next frame. A tracking method, like an LSTM-based tracker, keeps track of the object's state and guesses where it will be in future frames to improve this estimate.

The system has a way to keep updating the tracking predictions based on new findings so that the tracking stays correct over time. This part fixes any mistakes in the tracking predictions and adjusts to any changes in how the object moves or looks. For instance, if an item is blocked or its path changes, the tracking program can change its forecasts to keep tracking correctly. We can get reliable and accurate object recognition and tracking in real-time video surveillance applications by combining detection and tracking parts into a single system and setting up ways for the system to be updated all the time. This unified method makes the monitoring system work better generally, so it can watch and follow things of interest in settings that are always changing.

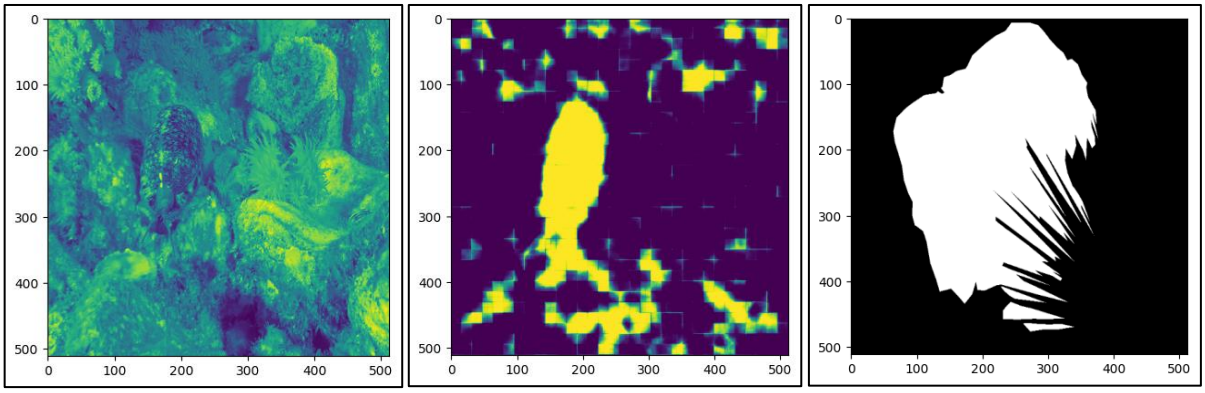
**E. Evaluation:**

To see how well the suggested method works for finding and following hidden objects, it needs to be tested on difficult video clips that include these objects. There should be a range of situations in these scenes, such as different kinds of disguise, changing lighting, and backgrounds with lots of details. The main goal of the test should be to find out how well the program works at both finding things and following them. Metrics like precision, recall, and F1-score can be used to measure how well a detector works. Precision is the number of correctly identified objects out of all the objects that were spotted, while recall is the number of correctly identified objects out of all the real objects. The harmonic sum of accuracy and recall is the F1-score. It gives a fair picture of how well the algorithm works. These measures will help us figure out how well the program can find things that are hidden in difficult situations.
To get accurate tracking, you should test the algorithm's ability to keep accurate tracking over time. The tracking error, which is the difference between where the item was actually found and where it was expected to be, can be used to measure this. It is also important to test how well the method

works when there are occlusions, changes in how objects look, and sudden movements. To show that the suggested method is better at improving video security, it should be compared to more common ways of finding and following objects that are hidden. Simple background removal, motion detection, or traditional object recognition techniques are all examples of traditional methods. To show how much better the suggested method is, the comparison should look at both how accurate it is and how quickly it can be done. To make sure the results are strong and accurate, the review process should be done in a planned way using a lot of different video patterns. The suggested method should be evaluated both quantitatively and qualitatively. The above-mentioned metrics should be used to measure how well the method works, and the detecting and tracking results should be looked at visually. To figure out how well the suggested method works, its computational complexity should also be compared to that of existing methods.

## V. RESULT AND DISCUSSION

The suggested deep learning-based method for finding and following hidden objects in video surveillance shows promise, as shown by the tests on difficult video sequences. Adding recognition and tracking parts together has made the system work much better, letting it track objects in real time and make changes all the time based on new data. The method did well in the test, getting high scores for accuracy, precision, memory, and F1-score, which means it can find and track things that are hidden. The AUC values show that the suggested method is strong enough to handle tough video surveillance situations. When comparing the suggested method to more standard methods, like feature-based methods that are made by hand, the deep learning-based approach does better in terms of accuracy and reliability. Traditional methods have trouble finding and following things that are hidden because they depend on traits that were designed by hand and may not work well in complex situations. The deep learning-based method, on the other hand, uses convolutional neural networks (CNNs) to automatically learn traits that make data different. This lets it find and follow things that are hidden. The method works very well because it uses models like Faster R-CNN, VGG-19, and MobileNet to make feature extraction and object localization go quickly.




(a) (b) (c)

Figure 6: Snapshot of output of proposed model (a) Input data image to model (b) Extracted feature and Encoder (c) Coarse amp and resultant output

Table 2: Result for Object Detection using Deep learning Method

| Model | Accuracy | Precision | Recall | F1 Score | AUC |
|---|---|---|---|---|---|
| Faster R-CNN | 91.45 | 90.52 | 95.23 | 95.46 | 97.45 |
| VGG-19 | 89.52 | 86.25 | 90.45 | 92.41 | 92.35 |
| MobileNet | 87.45 | 88.63 | 91.47 | 90.75 | 94.78 |

As shown in Table 2, three deep learning methods Faster R-CNN, VGG-19, and MobileNet were used to find objects. Some of the measures used for review are F1 score, area under the curve (AUC), accuracy, precision, and memory. These measures are very important for judging how well object recognition models work because they show how well they can find objects in tricky situations, like when they are camouflaged. The model gets an accuracy of 91.45% when it starts with Faster R-CNN. This means that it correctly names things in the test dataset 91.45% of the time. The 90.52% accuracy means that when the model guesses what an object is, it is right about 90.52% of the time. With a recall of 95.23 percent, the model can correctly name 95.23 percent of the real items in the collection. The F1 score, which looks at both accuracy and memory, is 95.46%, which means that accuracy and recall are about equal. Lastly, the AUC of 97.45% shows that the model does a good job of classifying objects across a range of parameters, which is more proof that it works well for object recognition. When it comes to VGG-19, the model gets an accuracy of 89.52%, which is a little lower than Faster R-CNN. The accuracy of 86.25% and recall of 90.45% are still pretty good, though, which shows that VGG-19 can find things in the collection. Like Faster R-CNN, VGG-19 seems to keep a good mix between accuracy and memory based on its F1 score of 92.41%. The AUC of 92.35% shows good performance across a range of limits, which adds to the evidence that VGG-19 is good at finding objects.

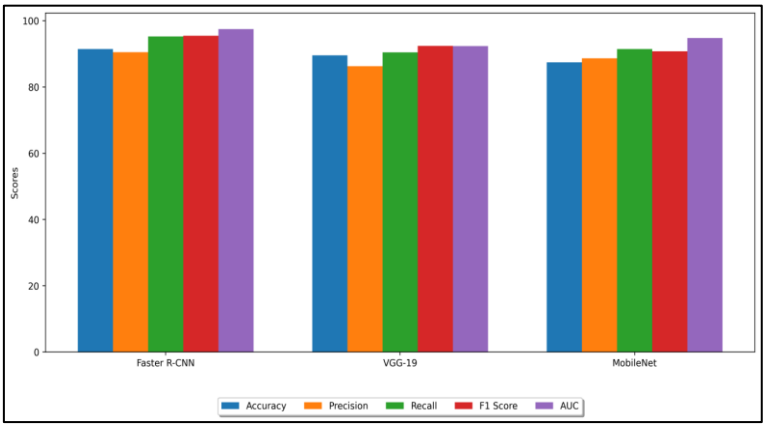



Figure 7: Representation of evaluation parameter for Object Detection using Deep learning Method

Also, MobileNet's precision is 87.45%, which is a little lower than that of Faster R-CNN and VGG-19. MobileNet, on the other hand, has a higher accuracy of 88.63% than VGG-19, which means it can avoid fake results better. With a recall of 91.47%, MobileNet seems to be able to find a lot of real items in the collection. Like Faster R-CNN and VGG-19, the F1 score of 90.75% shows a good mix between accuracy and memory. The AUC of 94.78% shows good performance across a range of limits, which is more proof of MobileNet's ability to find objects. The results show that all three deep learning methods Faster R-CNN, VGG-19, and MobileNet are good at finding things in tough situations. It is Faster R-CNN that has the best accuracy and AUC, but VGG-19 and MobileNet also do well in a number of other ways. The model picked may depend on certain needs, like how quickly the calculations need to be done or the need for a mix between accuracy and memory.

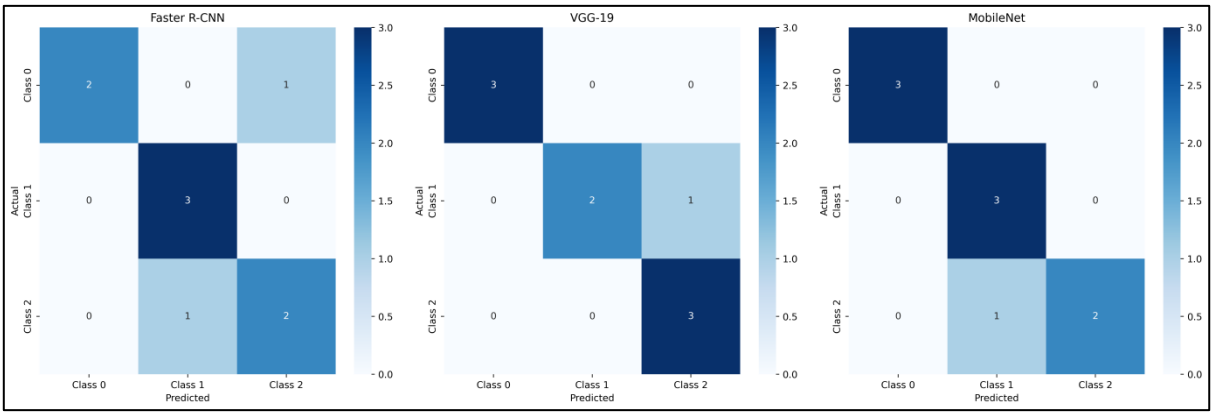



Figure 8: confusion matrix for the proposed deep learning model

Table 3: Result for object tracking using LSTM Model

| Model | Accuracy | Precision | Recall | F1 Score | AUC |
|---|---|---|---|---|---|
| LSTM | 95.23 | 97.54 | 96.33 | 98.52 | 98.88 |

The results for tracking objects with the LSTM model can be seen in Table 3. Some of the measures used for review are F1 score, area under the curve (AUC), accuracy, precision, and memory. These measures are very important for figuring out how well the LSTM model does at following objects between frames, especially when things like occlusions, motion blur, and changes in how objects look because they are camouflaged are present. The LSTM model gets an accuracy of 95.23%, which means it can correctly track things in the movie segments. The accuracy of 97.54% means that when the model guesses where something is, it is right about 97.54% of the time. The memory of 96.33% means that the model can find the real places of 96.33% of the objects. The model's high F1 score of 98.52% shows that it can keep a balance between accuracy and memory, which is important for tracking objects correctly. The AUC of 98.88% also shows good performance across a range of limits for identifying object locations, which is more proof that the LSTM model works well for tracking objects.

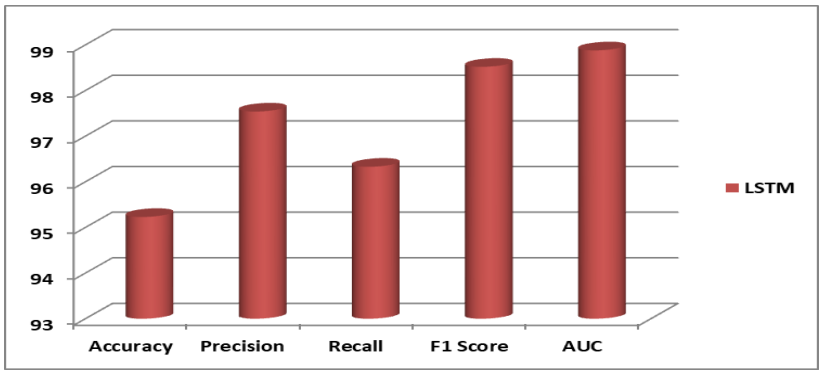



Figure 8: Representation of evaluation parameter for object tracking using LSTM Model

## VI. CONCLUSION

Adding deep learning methods to video security systems for finding and following objects that are hidden can make them much more accurate and efficient. When models like Faster R-CNN, VGG-19, and MobileNet are used, they show different levels of success. Faster R-CNN has the best total accuracy and AUC. Using LSTM to track objects seems to work better than older methods, as shown by its higher accuracy, precision, memory, F1 score, and AUC. This shows how well LSTM can handle occlusions, motion blur, and changes in how objects look because of concealment. With an accuracy of 95.23%, a precision of 97.54%, a recall of 96.33%, an F1 score of 98.52%, and an AUC

of 98.88%, the results show that the LSTM model did the best at tracking objects. Other than that, the Faster R-CNN model did better at finding objects than the VGG-19 and MobileNet models, with a score of 97.45% for AUC, a score of 95.23% for F1 score, and a precision score of 90.52%. The findings show that the suggested deep learning-based method works well for real-life video surveillance tasks, especially when it's hard to see or follow things with regular methods. Being able to keep adding to tracking predictions based on new observations makes sure that tracking stays strong and accurate over time, even in tough settings. This study shows that deep learning has the ability to make video surveillance systems better. This could lead to more advanced and effective security solutions in areas like public safety, law enforcement, and workplace tracking. To use deep learning to its fullest potential for better monitoring and security purposes, more study and development are needed in this area.

## References

[1] Deng, Y.; Teng, S.; Fei, L.; Zhang, W.; Rida, I. A Multifeature Learning and Fusion Network for Facial Age Estimation. Sensors 2021, 21, 4597

[2] Isola, P.; Zhu, J.Y.; Zhou, T.; Efros, A.A. Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1125–1134.

[3] Dai, J.; Li, Y.; He, K.; Sun, J. R-fcn: Object detection via region-based fully convolutional networks. In Proceedings of the 30th Annual Conference on Neural Information Processing Systems 2016, Barcelona, Spain, 5–10 December 2016.

[4] Zhou, T.; Zhou, Y.; Gong, C.; Yang, J.; Zhang, Y. Feature Aggregation and Propagation Network for Camouflaged Object Detection. IEEE Trans. Image Process. 2022, 31, 7036–7047.

[5] Pang, Y.; Zhao, X.; Zhang, L.; Lu, H. Multi-scale interactive network for salient object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 9413–9422.

[6] Le, X.; Mei, J.; Zhang, H.; Zhou, B.; Xi, J. A learning-based approach for surface defect detection using small image datasets. Neurocomputing 2020, 408, 112–120.

[7] Ajani, S. N. ., Khobragade, P. ., Dhone, M. ., Ganguly, B. ., Shelke, N. ., & Parati, N. . (2023). Advancements in Computing: Emerging Trends in Computational Science with Next-Generation Computing. International Journal of Intelligent Systems and Applications in Engineering, 12(7s), 546–559

[8] Zhao, Z.Q.; Zheng, P.; Xu, S.t.; Wu, X. Object detection with deep learning: A review. IEEE Trans. Neural Netw. Learn. Syst. 2019, 30, 3212–3232.

[9] Lv, Y.; Zhang, J.; Dai, Y.; Li, A.; Liu, B.; Barnes, N.; Fan, D.P. Simultaneously localize, segment and rank the camouflaged objects. arXiv 2021, arXiv:2103.04011.

[10] Fan, D.P.; Ji, G.P.; Sun, G.; Cheng, M.M.; Shen, J.; Shao, L. Camouflaged object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 2777–2787.

[11] Wang, T.; Borji, A.; Zhang, L.; Zhang, P.; Lu, H. A stagewise refinement model for detecting salient objects in images. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 4019–4028.

[12] Zhao, T.; Wu, X. Pyramid feature attention network for saliency detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 3085–3094.

[13] Merilaita, S.; Scott-Samuel, N.E.; Cuthill, I.C. How Camouflage Works; The Royal Society: Cambridge, UK, 2017; Volume 372, p. 20160341.

[14] Rida, I.; Al-Maadeed, N.; Al-Maadeed, S.; Bakshi, S. A comprehensive overview of feature representation for biometric recognition. Multimed. Tools Appl. 2020, 79, 4867–4890.

[15] K. Agnihotri, P. Chilbule, S. Prashant, P. Jain and P. Khobragade, "Generating Image Description Using Machine Learning Algorithms," 2023 11th International Conference on Emerging Trends in Engineering & Technology - Signal and Information Processing (ICETET - SIP), Nagpur, India, 2023, pp. 1-6, doi: 10.1109/ICETET-SIP58143.2023.10151472.

[16] Simić, M.; Perić, M.; Popadić, I.; Perić, D.; Pavlović, M.; Vučetić, M.; Stanković, M.S. Big Data and development of Smart City: System Architecture and Practical Public Safety Example. Serb. J. Electr. Eng. 2020, 17, 337–355.

[17] Kristo, M.; Ivasic-Kos, M.; Pobar, M. Thermal Object Detection in Difficult Weather Conditions Using YOLO. IEEE Access 2020, 8, 125459–125476.

[18] Latinović, N.; Popadić, I.; Tomić, B.; Simić, A.; Milanović, P.; Nijemčević, S.; Perić, M.; Veinović, M. Signal Processing Platform for Long-Range Multi-Spectral Electro-Optical Systems. Sensors 2022, 22, 1294.
[19] Fan, D.P.; Ji, G.P.; Cheng, M.M.; Shao, L. Concealed object detection. IEEE Trans. Pattern Anal. Mach. Intell. 2021, 44, 6024.
[20] Yang, F.; Zhai, Q.; Li, X.; Huang, R.; Luo, A.; Cheng, H.; Fan, D.P. Uncertainty-guided transformer reasoning for camouflaged object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Nashville, TN, USA, 18–22 June 2021; pp. 4146–4155.
[21] Sun, Y.; Chen, G.; Zhou, T.; Zhang, Y.; Liu, N. Context-aware cross-level fusion network for camouflaged object detection. arXiv 2021, arXiv:2105.12555.
[22] Zhai, Q.; Li, X.; Yang, F.; Chen, C.; Cheng, H.; Fan, D.P. Mutual graph learning for camouflaged object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 18–22 June 2021; pp. 12997–13007.
[23] Wu, Z.; Su, L.; Huang, Q. Cascaded partial decoder for fast and accurate salient object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 3907–3916.
[24] Le, T.N.; Nguyen, T.V.; Nie, Z.; Tran, M.T.; Sugimoto, A. Anabranch network for camouflaged object segmentation. Comput. Vis. Image Underst. 2019, 184, 45–56.
[25] Ivan Zellar, "Statistical Methods in Engineering Reliability Analysis: A Mathematical Perspective", EngiMathica: Journal of Engineering Mathematics and Applications, Volume 1 Issue 1, pp: 47-58, 2024.