



Article

Camera-Based Crime Behavior Detection and Classification

Jerry Gao ^{1,2,†} , Jingwen Shi ¹, Priyanka Balla ¹, Akshata Sheshgiri ¹, Bocheng Zhang ³, Hailong Yu ³
and Yunyun Yang ^{3,*} 

¹ Department of Computer Engineering, San Jose State University, San Jose, CA 95192, USA; jerry.gao@sjsu.edu (J.G.); jingwenshi@sjsu.edu (J.S.); priyankaalla@sjsu.edu (P.B.); akshatasheshgiri@sjsu.edu (A.S.)

² Department of Applied Data Science, San Jose State University, San Jose, CA 95192, USA

³ College of Electrical and Power Engineering, Taiyuan University of Technology, Taiyuan 030024, China; zhangbocheng1966@link.tyut.edu.cn (B.Z.); yuhailong0361@link.tyut.edu.cn (H.Y.)

* Correspondence: yangyunyun@tyut.edu.cn

† These authors contributed equally to this work.

Abstract: Increasing numbers of public and private locations now have surveillance cameras installed to make those areas more secure. Even though many organizations still hire someone to monitor the cameras, the person hired is more likely to miss some unexpected events in the video feeds because of human error. Several researchers have worked on surveillance data and have presented a number of approaches for automatically detecting aberrant events. To keep track of all the video data that accumulate, a supervisor is often required. To analyze the video data automatically, we recommend using neural networks to identify the crimes happening in the real world. Through our approach, it will be easier for police agencies to discover and assess criminal activity more quickly using our method, which will reduce the burden on their staff. In this paper, we aim to provide anomaly detection using surveillance videos as input specifically for the crimes of arson, burglary, stealing, and vandalism. It will provide an efficient and adaptable crime-detection system if integrated across the smart city infrastructure. In our project, we trained multiple accurate deep learning models for object detection and crime classification for arson, burglary and vandalism. For arson, the videos were trained using YOLOv5. Similarly for burglary and vandalism, we trained using YOLOv7 and YOLOv6, respectively. When the models were compared, YOLOv7 performed better with the highest mAP of 87. In this, we could not compare the model's performance based on crime type because all the datasets for each crime type varied. So, for arson YOLOv5 performed well with 80% mAP and for vandalism, YOLOv6 performed well with 86% mAP. This paper designed an automatic identification of crime types based on camera or surveillance video in the absence of a monitoring person, and alerts registered users about crimes such as arson, burglary, and vandalism through an SMS service. To detect the object of the crime in the video, we trained five different machine learning models: Improved YOLOv5 for arson, Faster RCNN and YOLOv7 for burglary, and SSD MobileNet and YOLOv6 for vandalism. Other than improved models, we innovated by building ensemble models of all three crime types. The main aim of the project is to provide security to the society without human involvement and make affordable surveillance cameras to detect and classify crimes. In addition, we implemented the Web system design using the built package in Python, which is Gradio. This helps the registered user of the Twilio communication tool to receive alert messages when any suspicious activity happens around their communities.



Citation: Gao, J.; Shi, J.; Balla, P.; Sheshgiri, A.; Zhang, B.; Yu, H.; Yang, Y. Camera-Based Crime Behavior Detection and Classification. *Smart Cities* **2024**, *7*, 1169–1198. <https://doi.org/10.3390/smartcities7030050>

Academic Editor: Pierluigi Siano

Received: 26 February 2024

Revised: 4 May 2024

Accepted: 6 May 2024

Published: 19 May 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: object detection; crime classification; deep learning; arson ; burglary; vandalism; crime; Twilio; Gradio

1. Introduction

Crime-prone locations have become more challenging because of a recent increase in the population in urban areas [1]. There has been an upsurge in crime and insecurity

because of this lack of control. Innovative solutions to these issues are now possible because of the development of smart city infrastructure [2]. It is possible to identify illegal activities with the help of an auto regressive model and behavioral recognition techniques [3]. Traditional crime-solving methods are ineffective in the current climate because they are too time-consuming and inefficient to keep up with the escalating crime rate [4]. According to the literature survey, the accuracy evaluation of crime types can be obtained in different ways, as shown in Figure 1.

Training Data	Pattern recognition CNN	Time Series RNN	Models	Accuracy	Evaluation	Crime type
UCF101, ActivityNet, Kinetics-400, YouTube8M, UCFCrime, CityScene	Y	Y	3DCNN, 2DCNN + TSM	89% 90%	Accuracy	Real time crime anomalies
Image-Net Dataset	Y	Y	CNN-LSTM	98%	Training accuracy and testing accuracy	Vandalism
AI Hub	Y	Y	RestNet50 + LSTM, 3DRestNet, ID3	59% 75% 93%	Train error and test error	Real time crime anomalies
CAVIAR, KTH, Campus videos, YouTube videos	Y	Y	3DCNN (DrakNet-19, RestNet-101, RestNet-152, DarkNet-53)	74% 77% 77.6% 77.2%	Accuracy	Real time crime anomalies
Urban Crime Videos	Y	Y	DCNN + RNN (HDL)		Accuracy based on ranked video frames	Real time crime anomalies
CCTV infrastructure installed, camera videos	Y	N	YOLO, Fast-RCNN	78% 64%	measured by mean average precision	Real time crime anomalies
CCTV videos	Y	Y	RCNN, (RPN)	99%	Accuracy	Arson
UCF Crime Dataset	Y	Y	3DCNN	89%	SGD optimizer ROC curve	Arson

Figure 1. Literature survey.

The technology survey shows the advantages, disadvantages, and best use cases. See Figure 2.

Model	Concept	Advantages	Disadvantages	Best Use Case
ResNet-101	ResNet-101 is a CNN architecture model which has 101 layers; for every few convolution layers, ResNet-101 will build a skip connection between them.	Requires less memory size, easy to optimize Provides high accuracy.	When the depth of ResNet-101 increases it might cause a higher training error.	Image classification, object detection.
SDD	The SDD architecture built based on VGG-16 network architecture, additional convolutional layers were added rather than fully connected layers with multibox method.	Provides higher accuracy when it has more multibox.	Performs badly on small objects since it might not cross the feature maps; mixed objects by similar categories.	Real-time object detection.
Inception-v3	Inception-v3 is a CNN architecture model which uses RMSProp optimizer, label smoothing, factorized 7 by 7 convolutions and auxiliary Classifiers to do the classification.	Provides good performance; higher efficiency and deeper network; less expensive.	Performance might have big contrast use to different data and use cases.	Object detection, image classification
YOLO-V3	YOLO-V3 has 53 layers originally and a total of 106 layers of fully convolutional architecture. 1 by 1 kernel shape applied to 3 different sizes of feature maps at 3 different places, and given strides of 32, 16, 8 respectively.	Prevent loss of the low-level features; Fast training speed, efficient for any object detection problems.	Not ideal for large datasets which the dataset is hard to collect.	Real-time object detection, image classification.

Figure 2. Technology survey.

For the shortcomings of existing technology, it would alleviate the pressure on police and aid in crime prevention if we could devise methods for predicting crimes, in detail, before they occurs, or devise a “machine” [5] that can help police officers. We propose the use of machine learning and computer vision methods [6] and approaches to accomplish this.

This paper presents a thorough evaluation of the field, including topics such as object recognition, group research, and, finally, action detection in video frames or clips. In this effort, we are focusing on the detection of four specific types of criminal activity: arson, theft,

burglary, and vandalism. At the outset, the video inputs were reduced, and the videos were annotated [7]. Video annotation is the process of manually labeling and classifying video footage to teach computer vision algorithms how to find and identify objects. Marking items in a video, as opposed to an image, frame by frame, enables machine learning models to identify the objects in the video. The spatial and temporal characteristics [8] of video must both be accounted for in an appropriate video index. To achieve this, video is segmented into shots, and then the most relevant frames are extracted for use in indexing and retrieval. This class includes scenes or video clips that have been frozen in time, such as observable motion and the object's fixed characteristics. Features are extracted from the video data. The information was then separated into three groups: the training set, the test set, and the validation set. To better comprehend the patterns [9] and provide more accurate results when tested with the test datasets, the deep learning models were trained with the training datasets of corresponding crimes (arson, stealing, burglary, and vandalism). All the videos were turned into frames using Roboflow to perform the further preprocessing methods. We used YOLOv5 for arson objects, YOLOv7 for burglary objects, and YOLOv6 for vandalism object detection [10]. Furthermore, we implemented OpenCV for classification of the crimes based on the objects that get detected in the video input. To gauge the efficacy of the outcomes, we conducted evaluation measures. We implemented a user interface using Gradio, which is easy to use and verify the surveillance videos to check the crime behavior. Meanwhile, the aim of the project is to implement a robust automated intelligence platform that detects and classifies specific crime types (arson, burglary, and vandalism) within Homeowner Association (HOA) communities (HOA communities are official groups of residents that preside over a community), parking lots, and apartment buildings from camera-based videos and surveillance videos [11]. Structure: The rest of this article is organized as follows. Data collection and data process are described in Section 2, with details on the training and preparation of test data. Section 3 describes the method for selecting the model. In Section 4, data analysis and intelligent system design are presented. Section 5 is the evaluation of the performance of the intelligent system and the display of the results. Section 6 discusses the results of this study in light of other similar studies.

2. Data Engineering

2.1. Data Process

To collect the data for this paper, we searched through a large number of Google-accessible websites that provided us with the opportunity to collect crime films and datasets that include a classification of various types of crime. There was a grand total of six distinct varieties of footage that needed to be gathered to adequately demonstrate our goal. All the different categories of crime this paper hoped to collect are detailed in Table 1 below.

Table 1. Different types of crime videos.

Crime	Places
Arson	Parking lots.
Arson	Neighborhoods, houses, streets, etc.
Burglary	Parking lots.
Burglary	Neighborhoods, houses, vehicles, streets, etc.
Vandalism	Parking lots.
Vandalism	Neighborhoods, houses, streets, etc.

Figure 3 depicts our project's complete data process lifetime, beginning with the collection of the raw video information and ending with the final product. Data preprocessing, data transformation, and data preparation are the major components. As part of the data preprocessing phase, we manually clipped films for each crime category and then used those segments to build frames. Since most crime videos we received were only a few minutes long, and the crime itself only occurred during a brief window of time, we needed to trim the videos down to just the relevant parts in order to maximize the effectiveness of

our training, which also entailed creating frames for every trimmed video. Frame resizing, data augmentation for greater lighting and perspective of frames, and annotating frames are all examples of data transformation that are used as input to a model. Separating raw data into three distinct sets—a training dataset, a validation dataset, and a testing dataset—was the next step in the data preparation process. Three Google drive folders were created: one for the training data, one for the validation data, and one for the testing data. A total of 80% of the training data was utilized to train our deep learning models, with 15% for validation, and 5% for testing.

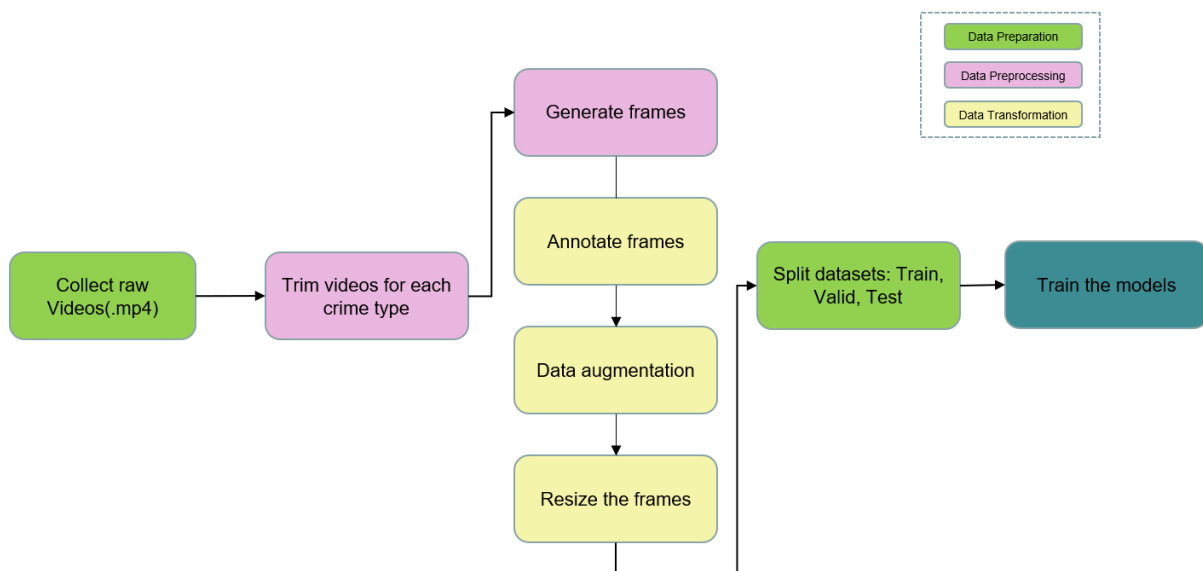


Figure 3. Data Preprocessing.

Data augmentation can assist in enhancing the likelihood of accurate detection by rotating image frames to capture targets from various angles, in addition to adjusting the brightness of frames to optimize detection capabilities. The main focus of our detection was the criminal behaviors, but due to the limitation of the videos, some crime behaviors were not easily observed; our main goal was therefore redirected to the detection of some common suspicious actions, such as human running or hovering; the critical information tends to be obtained by focusing on a specific person.

2.2. Data Collection

We collected anomalous and normal videos from two different sources for Burglary, Arson, and Vandalism. Table 2 lists all the raw video data statistics for each dataset with crime type, quantities, and whether the raw datasets were annotated or not. Each following paragraph will detail parameters, sources, and samples from raw datasets. Each video is defined with specific crime types so that we can identify each crime type with the respective folder whereas normal videos are available in the normal folder. We select each video that helps define our crime type and discard the other videos. We trim selected videos specific to the crime event and normal event and will be maintained in the particular crime and normal folders. Once we trim the video we generate frames and make them available to annotate the objects. Some videos will provide a good amount of frames whereas others might be based on crime events in videos. Once we label the objects for particular crime types, we split the data into train, test, and validate folders for each crime type. Arson: 80%, 5%, 15%, Burglary: 80%, 5%, 15% and Vandalism: 80%, 5%, 15%. Since we will be testing on videos rather than test video frames, we are considering only 5% of video frames. However, we will be testing on different scenarios including day and night times.

Table 2. Different types of crime videos.

Dataset	Crime Type	Untrimmed Videos	Total Videos	Raw Video Frames	Data Augmentation	Total Frames	Manual Annotation
UCF [8]	Arson	60		3000	5500		
YouTube	Arson	40	140	1640	1500	8050	YES
Storyblocks [12]	Arson	40		1000	1050		
UCF [8]	Burglary	110		4000	5500		
YouTube	Burglary	50	200	1200	2100	9100	YES
Storyblocks [12]	Burglary	40		1000	1500		
UCF [8]	Vandalism	50		3060	4405		
YouTube	Vandalism	80	170	1460	2945	9050	YES
Storyblocks [12]	Vandalism	40		1000	1700		

Source 1: UCF crime dataset [8] is the largest open dataset for real-time surveillance crime videos and contains 14 different crime types. Based on the project focus, we collected burglary, arson, vandalism, etc. We have 5500 arson, 4405 vandalism and 5500 burglary video frames. The purpose of collecting data from UCF [8] varied across different parameters, such as high-resolution videos, captured from different surveillance cameras, night and day visions camera models, videos from different camera angles, etc.

2.3. Data Preprocessing

Since the data were videos, only minimal data preprocessing was required. The steps are shown below to preprocess the models for training and achieve good results:

- (1) Step 1 involves data cleaning by selecting good resolution videos in each crime type to get good results.
- (2) Step 2 is finding an anomalous event in a video and defining the objective of abnormal activity.
- (3) Step 3 is video trimming for suspicious and normal events using inbuilt video trimming software tools from Mac and Windows.
- (4) Step 4 is generating frames from each video by developing Python code.
- (5) When extracting frames, each frame name will be annotated. We created annotations for each frame using Roboflow and assigned class names for each target object.
- (6) The generated video frame dimensions might be different but we changed the dimensions of the extracted frames to different heights, weights, and widths as input size for the models in the data transformation section.
- (7) To increase our training data, data augmentation was applied: horizontal and vertical flips, sheer, rotating, and zooming for existing frames that provide different camera views of crime activities.

2.4. Data Transformation

2.4.1. Data Transformation—Resizing the Frames

The videos that were obtained from the different data sources like UCF, HMDB51, and YouTube were converted to frames in the data preprocessing step. The frames which are the output of the data preprocessing step, were of the size 320×240 . Transforming data from one pattern to another, often from one source system to another, is known as data re-formatting or transformation. There are several common data management and integration jobs that include data transformation. In this project, we used different models to train the data and, accordingly, we resized the frames to 224×224 size. When you resize an image, you can make it smaller or larger without removing any of the original content. File size and quality are often impacted by resizing an image because it changes its proportions. We have done the preprocessing steps and resizing of the frames using the Python programming language. We imported the libraries like cv2, glob, os, and time to accomplish the resizing of the frames. We collected the list of videos in a folder and passed it as an input to the main method and predefined the output folder which helps

us to store all the frames collectively. The frames were generated for one video at a time and the numbering started from 1 along with the video name as the labeling of the frame. Furthermore, for another video, the frames got generated automatically and the numbering started again from 1 with the respective video name as the labeling, and similarly for all the videos in the folder. We collected the data with respect to crime types in different folders which was helpful to train the data. While training, we planned to resize the data according to the models' input shape requirements. We did this entire process using a Python script in Jupyter Notebook.

2.4.2. Data Augmentation

Data augmentation is accomplished by incorporating information from both various sources into the base data. It is possible to improve data quality and reduce the amount of manual intervention necessary to produce relevant information and insight from business data by enhancing the data's augmentation. In order to increase the model's performance and generalizability, image data augmentation was performed. We utilized Jupyter Notebook as a tool and the Python programming language to perform the operation. This operation was supported via Roboflow. Different forms like horizontal/vertical shift, the brightness of images, and random rotation of images were applied. We loaded the frames one after the other, which needed to be augmented and they were converted to an array. By creating an image data augmentation generator and preparing the iterator, we generated samples and then plotted them. We applied different methods to the frames, as mentioned below.

2.5. Data Preparation

Based on the collected videos and the frames extracted, annotated, and organized in accordance with the offenses, we divided the final datasets into three groups: training, validation, and test (arson, burglary, and vandalism). The training tools were modified to accommodate the models. The prediction error of a candidate model was estimated using the validation set. After a final model was chosen, generalization errors were measured with a test set. Roboflow was used to organize the data into distinct directories and file types, with the annotation files for each frame clearly labeled. Depending on the kind of models being used, the file extensions was .csv,.xml, or.txt. All of the input files were mixed up before being divided into train, validation, and test dataset folders,so that each set of data was truly random. By defining the ratio of 70%-15%-15% training, validation, and testing, we are able to achieve our aims with the help of the package module, which distributes the dataset randomly across several output folders. We divided the data, which includes both crime footage and regular films, in the ways stated below for the various types of crime.

2.6. Data Statistics

In the data collection section, Table 3 depicts the crime type and umber of videos collected in the raw data: burglary: 200 videos, arson: 140 videos, and vandalism: 170 videos with the file format (.mp4), annotated or not from different sources that are publicly available. We have also presented video samples from each source for all the crimes by defining each sample frame about the anomalous event.

Table 3. Prepared video count for each crime type and normal type.

Crime Type	Crime Video Quantity (mp4)
Arson	140
Burglary	200
Vandalism	170
Normal	60

The following Figure 4a showcases the frame count for each crime type after performing data augmentation on the above frames.

The following Figure 4b showcases the total raw dataset video count of crime types and normal events. For normal events, we only chose 60 videos, which were lengthy and processed frames.

The following Figure 4c showcases the raw frame count for each crime type for the trimmed videos.

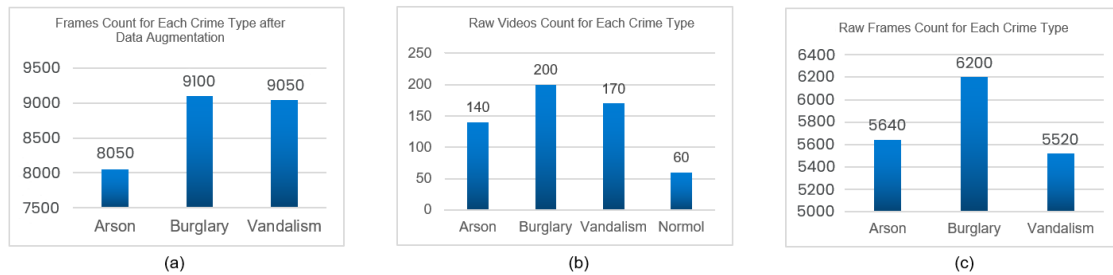


Figure 4. (a) Frame count after data augmentation, (b) total raw datasets video count for crime types and normal events, and (c) raw frame count for each crime type.

Table 4 lists the number of frames for each crime type with augmentation and split. Each video generates frames based on the crime event in the video. Since each video is trimmed to the required crime event, the number of frames for each video is different. In the data transformation, we presented video frames conversion from the raw dimension (320, 240) to the required dimension (224, 224). However, the number of frames remains the same.

Table 4. Summary of raw, preprocessed, transformed, and split data.

Crime Type	Source	Raw Frames	After Augmentation (Total Frames)	Train (80%)	Valid (15%)	Test (5%)
Arson	UCF, Youtube, Story Blocks	5640	8050	6440	1208	402
Burglary	UCF, Youtube, Story Blocks	6200	9100	7280	1365	455
Vandalism	UCF, Youtube, Story Blocks	5520	9050	7140	1524	386

Data Analytics Results: Below are the figures for the object count for each of our crime types. Figure 5a lists the object count for arson, Figure 5b lists the object count for burglary, and Figure 5c lists the object count for vandalism.

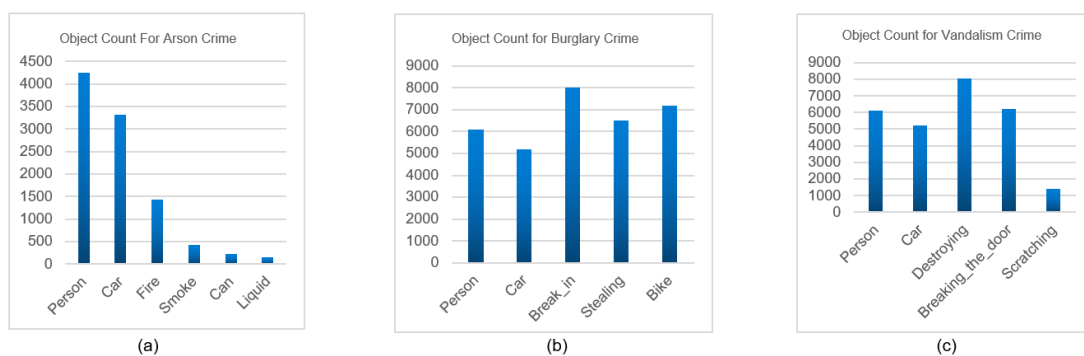


Figure 5. Total raw datasets video count.

The following Figure 6 displays the chart for the data split for each crime type with the proportion (train 80%, test 5%, and valid 15%).

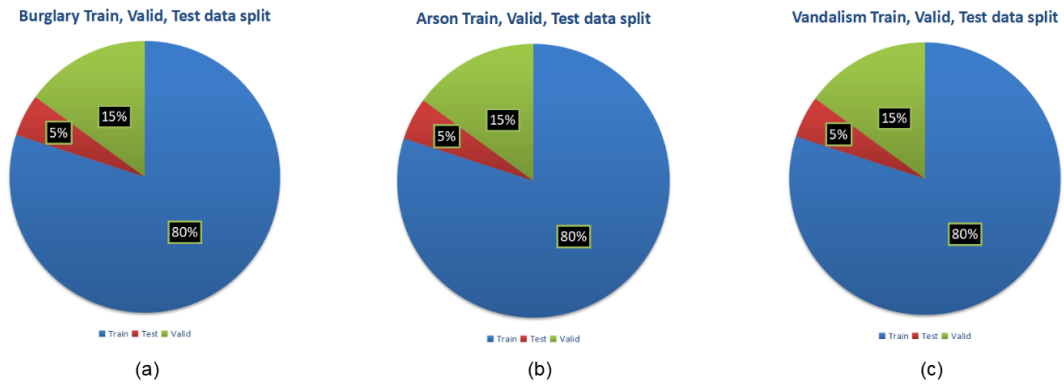


Figure 6. (a) Burglary train, valid, test data split, (b) arson train, valid, test data split, and (c) vandalism train, valid, test data split.

3. Model Development

3.1. Model Proposals

Improved YOLOv5: YOLO, also known as you only look once, is an algorithm of object detection [13]; it redefines object detection as a regression problem, applies a single convolutional neural network (CNN) to the entire image, divides the image into grids, and predicts class probabilities and bounding boxes for each grid. The YOLOv5 model is improved based on the YOLOv3 model [14]. There are four models: YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x. The YOLOv5 model consists of backbone network, neck, and head. The backbone network is composed of Focus, BottleneckCSP, and SSP networks including modules such as Focus, Conv convolution block, BottleneckCSP, and SSP, with an input image sized $640 \times 640 \times 3$; we sliced through the Focus module to reduce the height and width of the image, output the image size to 320×320 , and convert the height and width of the sliced image through Concat Integrate, increasing the number of channels of the input image to 64, then performing feature extraction on the integrated image to get a featured mAP (mean average precision), the SSP uses max pooling and aggregates feature mAPs through Concat. For the neck part, a bottleneck was used in order to improve network speed while ensuring accuracy. The head of YOLOv5 uses multi-scale feature mAPs for detection; large images are used to detect small targets, and small images are used to detect large targets, and finally, we obtained three scales of feature mAPs. There were only two types of crime detection for arson which were arson and normal, and prediction based on our self-made dataset, then we output the target as well as bounding box. To better extract the features from the input, the model was pre-trained by the COCO dataset, as our customized dataset applied transfer learning to predict and classify arson crime, replacing the default eighty classes to six classes. The backbone was the layers to extract input image features, and we froze the first ten layers of the backbone so the weights would not change during transfer learning, and change to smaller batch sizes in order to decrease the computation cost with faster training. The arson behavior has been identified using the Improved YOLOv5 Architecture, as depicted in Figure 7.

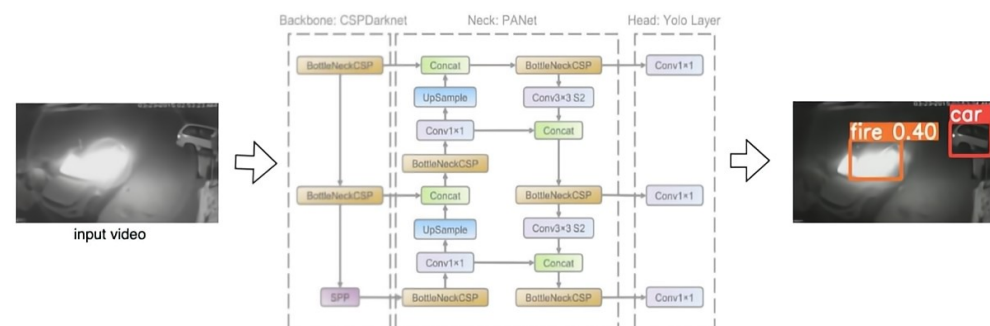


Figure 7. Improved YOLOv5 architecture.

3.1.1. Improved Faster RCNN ResNet101

Faster RCNN is a region-based object detection model that is widely used for image classification. Since our target is to detect the objects in images or videos, the model was used to detect the features in different regions of images and utilized to classify them as burglary and not burglary. Figure 8 shows that Faster RCNN architecture consists of two main components: region proposal network (RPN), and region of interest (ROI). Regions identified as objects/target labels within an image belong to the foreground class and are not identified as belonging to the background. To recognize these areas, RPN architecture introduces three main anchor boxes: 128×128 , 256×256 , and 512×512 to identify the variations, scale, size, and aspect ratios, 1:1, 2:1, and 1:2, of objects at each location in an image with a total of nine boxes on the RPN that assists in identifying different shapes of objects. When the anchor boxes intersect over union, if the threshold is greater than 0.5, those are labeled as foreground class and, otherwise, background class. This shared computation significantly reduces time cost and helps in detecting objects faster. In order to do this, RPN uses a CNN to extract those feature mAPs and then sends them to the ROI layer to flatten all the images to the same size.

To improve the model features extracted from input videos, Faster RCNN ResNet150, the pre-trained model on the COCO dataset, was trained on 2.5 million images with freeze convolutional base and without an output layer. We modified the last output layer with sigmoid activation and set class labels to five classes instead of eighty. The pre-trained ResNet101 model uses 101 convolutional layers, batch normalization, and relu activation functions in each block. Even though the model has 101 layers, it still failed to detect small objects, such as handy tools used to break into houses, cars, and vehicles. To resolve this issue, Feature Pyramid Extractor was utilized, which takes a single-scale image of an arbitrary size as input, and outputs proportionally sized feature mAPs at multiple levels, in a fully convolutional fashion. This process is independent of the backbone convolutional architectures and helps in learning and detecting tiny objects.

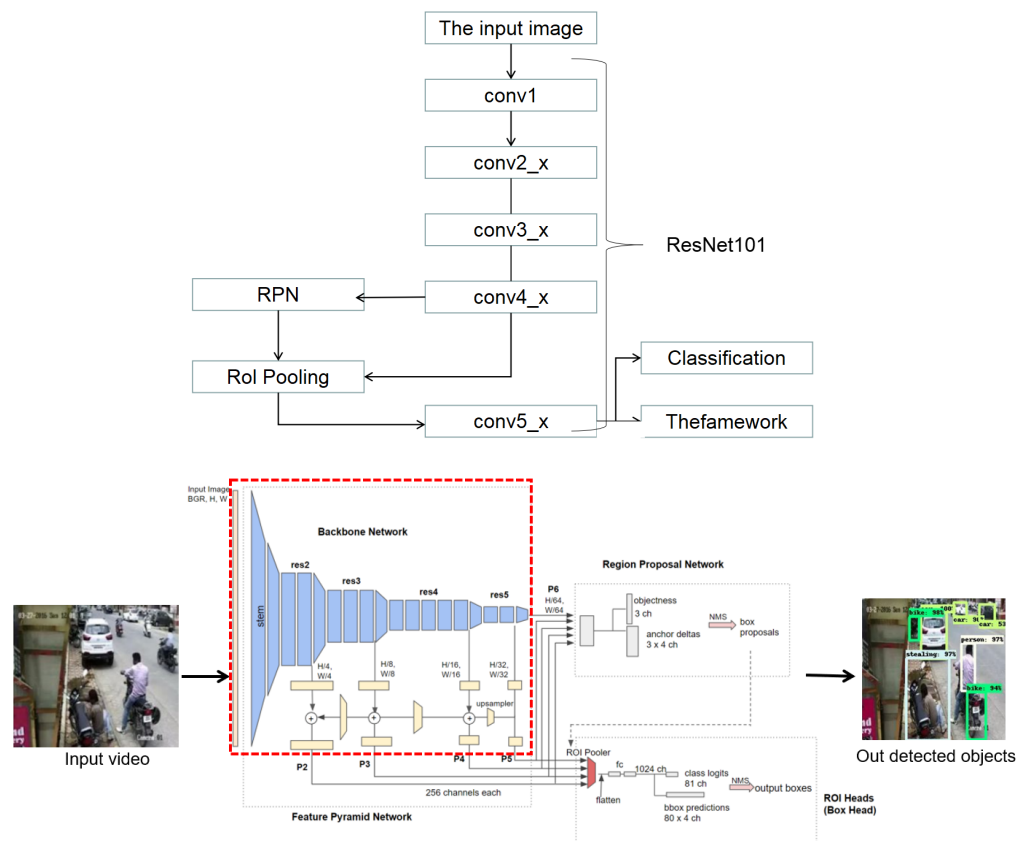


Figure 8. Improved Faster RCNN.

3.1.2. Improved YOLOv7

Yolo is an object detection model that is widely used for image classification. Since our target is to detect the objects in images or videos, the model was used to detect the features in different regions of images and utilized to classify them as burglary and not burglary. Figure 9 shows that the YOLOv7 architecture consists of four main components: backbone network, Feature Pyramid, neck, and head. The backbone network, composed of Focus, BottleneckCSP, and SSP networks including modules such as Focus, Conv convolution block, BottleneckCSP, and SSP, with an input image size of $640 \times 640 \times 3$, sliced through the Focus module to reduce the height and width of the image and output the image size to 320×320 , and converted the height and width of the sliced image through Concat Integrate, increased the number of channels of the input image to 64, then performed feature extraction on the integrated image to get a featured mAP; the SSP used max pooling and aggregated feature mAPs through Concat. The FPN solved the multi-scale problem that resolved issues in detecting small objects. For the neck part, a bottleneck was used in order to improve network speed while ensuring accuracy. The head of YOLOv7 used multi-scale feature mAPs for detection; larger images were employed for detecting smaller targets, whereas smaller images were utilized for identifying larger objects, and finally, we obtained three scales of feature mAPs. YOLOv7 also provides several model weights: YOLOv7, YOLOv7-tiny, YOLOv7-e6, YOLOv7-d6, YOLOv7-e6e, YOLOv7-w6, and YOLOv7x. To improve the model features extracted from input videos, YOLOv7, pre-trained on the COCO dataset, was trained on 2.5 million images with 80 classes with freeze convolutional base and without an output layer. We modified the last output layer with sigmoid activation and set class labels to five classes instead of eighty. The pre-trained darknet used a single convolution layer for entire images that divided an image into grids, and predicted each class probabilities and bounding boxes for each grid batch normalization, and relu activation functions in each block. Even though the model applied convolution layers efficiently, it still failed to detect small objects such as handy tools used to break in houses, cars, and vehicles. To resolve this issue, Feature Pyramid Extractor was utilize, which takes a single-scale image of an arbitrary size as input, and outputs proportionally sized feature mAPs at multiple levels, in a fully convolutional fashion. This process is independent of the backbone convolutional architectures and helps in learning and detecting tiny objects. Additionally, batch size was reduced to 16 to avoid out-of-memory issues, an increase in the number of steps to get higher precision and recall, and added more training data to avoid overfitting problems. Furthermore, transfer learning was done by freezing the last three layers to fine-tune the model.

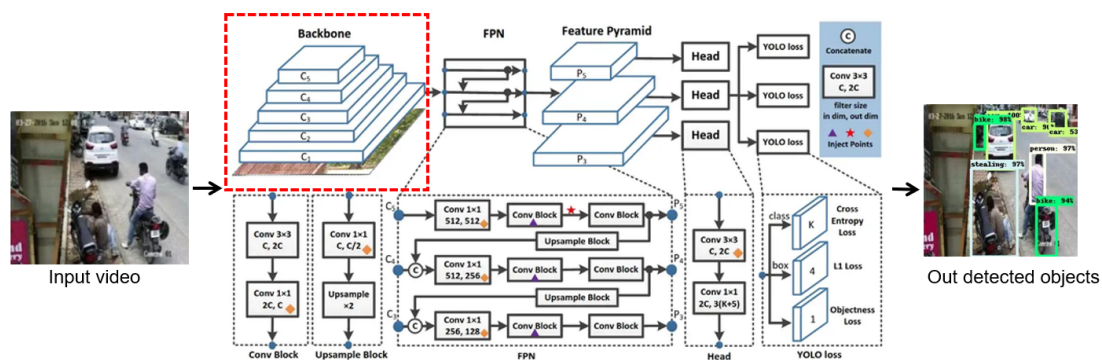


Figure 9. Improved YOLOv7.

3.1.3. Improved SSD Mobilenetv2

Real-time object detection [15] is SSD's (Single Shot detector) primary purpose. It has reduced wait time thanks to SSD's ability to bypass the regional proposal network. SSD employs some upgrades, such as a number of co-features and default boxes, to recoup the loss in precision. The enhancements allow SSD to achieve parity with the Faster R-CNN

in terms of accuracy while making use of images with lower resolution, hence increasing SSD's speed [16]. The model's name, "Single Shot detector", gives away a lot of its salient features. Unlike other models that must traverse the input image multiple times to acquire an output detection, the SSD model identifies the item in a single pass. The SSD model just needs one pass to detect objects, therefore it is faster. However, simultaneously, the SSD model appears to have remarkable detection performance. The SSD model clearly divides predictions by display size and generates predictions at many scales based on the sizes of the feature mAPs in order to attain high detection accuracy. These methods simplify end-to-end training and produce great accuracy, even with low-resolution input photos.

The SSD model consists of two components: the backbone model and the SSD head. To create the feature mAP, the backbone model employs a standard pre-trained image classification network. Here, only the retrieved feature mAPs remain after the model's final picture categorization layers have been omitted. The SSD head, which consists of many convolutional layers, is an extension of the main model. It produces bounding boxes over the object as an output. The numerous items in the image are identified by these convolutional layers. Sigmoid activation was added to the final output layer, and there are now just five classes instead of eighty. Each block in the Mobilenet model's inverted residual structure was trained using 53 convolutional layers, batch normalization, and relu activation functions. It got rid of the non-linearities in the thin layers. Despite the model's 53 layers, it was unable to recognize even relatively small things, such as personal tools, vehicles, and bicycles. To address this problem, Feature Pyramid Extractor, as seen in Figure 10, was used, which, given an input image of arbitrary scale, generates feature mAPs of progressively smaller sizes using a fully convolutional neural network. Learning and detecting small objects was aided by this method, which is decoupled from convolutional architectures used for the backbone. More training data were added to combat overfitting, and the batch size was decreased to eight. This allowed for increased precision and recall without running out of memory.

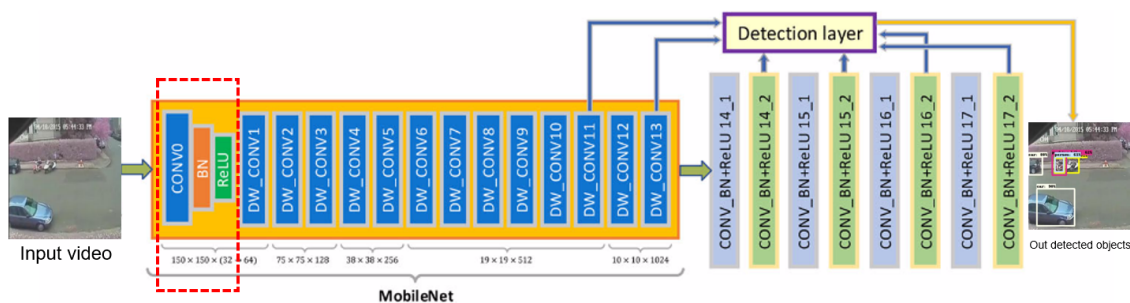


Figure 10. Improved SSD MobileNet architecture.

3.1.4. Improved YOLOv6

The single-stage object detection framework YOLOv6 is optimized for use in industrial settings thanks to its hardware-friendly, efficient design and top-notch performance. In terms of detection accuracy and inference speed, it is superior to YOLOv5, making it the optimal OS version of the YOLO architecture for real-world deployments. Although MT-YOLOv6 is the official name [17], YOLOv6 has been adopted by developers for the sake of brevity. The foundation of the model is the YOLO (you only look once) architecture, and the authors assert that it has various advantages over other models in the YOLO family thanks to these innovations and additions. PyTorch is the language of choice for this framework. The YOLOv6 object detector is often regarded as the most precise option available. The fact that the YOLOv6 Nano model obtained a mAP of 35.6% [18] on the COCO dataset is indicative of this. Additionally, it achieves frame rates of over 1200 FPS on a 32-bit NVIDIA Tesla T4 GPU. Among many other methods, the authors made significant modifications to the infrastructure, employed model quantization techniques, and utilized various augmentations to achieve significant impact on the results. In contrast to earlier YOLO architectures,

YOLOv6 uses an anchor-free approach to object identification Figure 11. By comparison, most anchor-based object detectors are 51% slower than YOLOv6. Because it uses three times fewer specified priors, this is feasible. In order to function, YOLOv6 relies on the EfficientRep framework, which is comprised of CSPStackRep, RepConv, RepBlock, and blocks. For both classification and box analysis, YOLOv6 uses VFL and DFL as loss functions. Iterating on the YOLO framework, YOLOv6 reworks the spine and neck to better fit the available hardware. The model includes a Rep-PAN Neck, as shown in Figure [19], and a new type of backbone called EfficientRep Backbone.

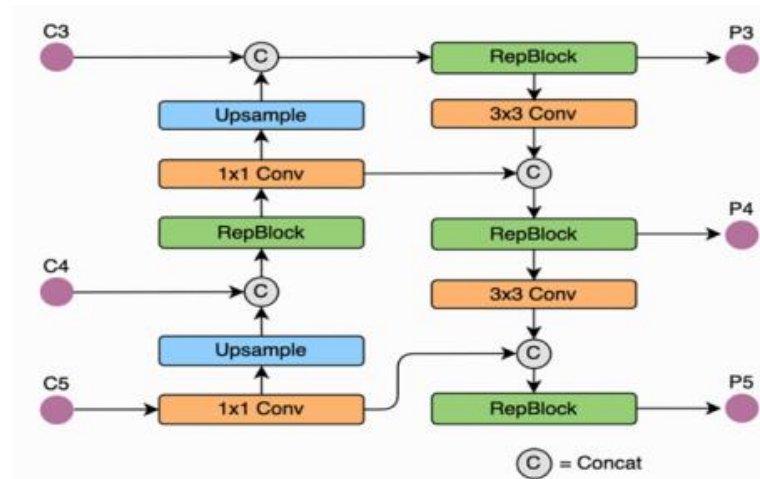


Figure 11. Rep-pan structure.

To improve the model, YOLOv6 pre-trained its model using the COCO dataset, which is trained on 2.5 million images with 80 classes using a frozen convolutional basis and without an output layer, in order to improve the model features derived from input videos. Sigmoid activation was added to the final output layer, and there are now just five classes instead of eighty. The importance of the backbone in feature extraction is crucial in any object detection network. The network’s neck and brain obtain these features. Since the backbone performs so much of the network’s processing, it is obviously crucial. Multi-branch networks, such as ResNets, have superior classification performance but are slower to infer. Linear networks, such as VGG, are substantially quicker due to their efficient 33 convolutions [18]. They are not as accurate, however, as ResNets and other networks built on the principle of residual connections. Consequently, the YOLOv6 models employ reparameterized backbones. Through the process of reparameterization, shown in Figure 12, the structure of the network is modified while it is learning and making predictions. The Feature Pyramid Extractor was used, which, given an input image of arbitrary scale, generates feature mAPs of progressively smaller sizes using a fully convolutional neural network [20]. Learning and detecting small objects was aided by this method, which is decoupled from convolutional architectures used for the backbone. More training data were added to combat overfitting, the batch size was increased to 48. This allowed for increased precision and recall without running out of memory [21].

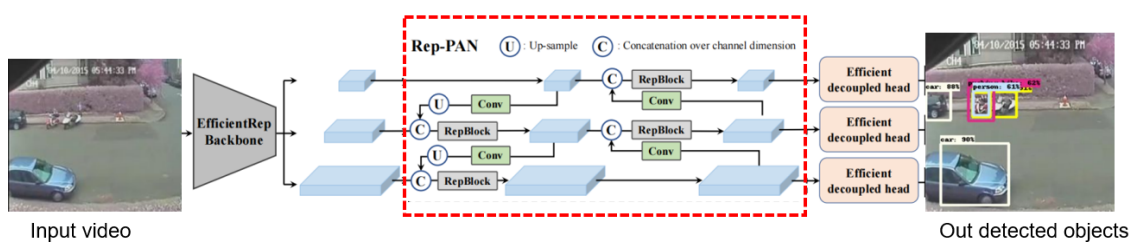


Figure 12. Improved YOLOv6 architecture.

3.1.5. Integrated Model

For the ensemble model we combined all of the crime object detection models, Improved YOLOv5, Improved YOLOv7, and Improved YOLOv6 for arson, burglary, and vandalism crime types, respectively. To get parallel results of all models at a single time, when we loaded an input video, open cv read each frame and sent that frame to all the three models for object detection. Then, based on detected crime or normal objects, the highest vote system was considered to eliminate other objects and classify the crime type. Figure 13 depicts the integrated model flow.

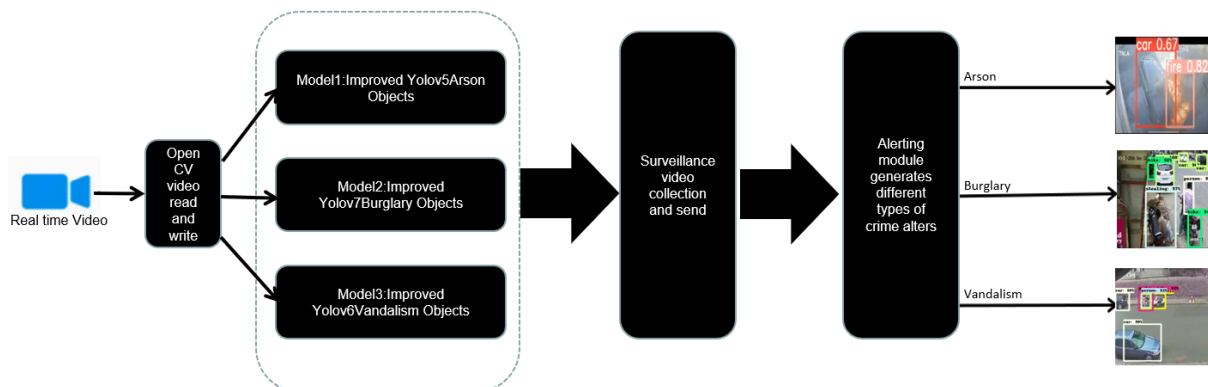


Figure 13. Smart crime watch model.

3.2. Model Support

Deep learning [2] platforms and data analytics throughout this project used Python to run through both Jupyter Notebook and Google Colab. Both of them are IDE web-based; when it comes to data security [22], Jupyter Notebook is considered safer than Google Colab; while Google Colab makes it more convenient and easier for us to collaborate with each other, its cloud-based collaboration is interactive and includes limited free computation power (GPU or TPU), which is not included in Jupyter Notebook. The Jupyter Notebook could run on our local computer or laptop and the files can be saved on the hard disk, but Google Colab runs on the Google server and our files are stored in the relevant Google Drive account. Google Colab enables faster training speed but, in the meanwhile, requests to pay for upgrades; we needed to pay attention to costs when training our model. Furthermore, for Jupyter Notebook, we needed to install all the required libraries based on what the requirement was, while in Google Colab we did not need to install these libraries, since almost all of the libraries are already pre-installed in Google Colab.

To execute deep learning models with their support tools for model development, we primarily used two Python packages: Tensorflow and Keras. Tensorflow is an end-to-end open-source platform for deep learning; it is a flexible tool that provides workflows with both high-level APIs and low-level APIs, and it is easy for model training, while Keras is a high-level neural network library that is built upon Tensorflow and it only provides high-level APIs, but Keras is simple and consistent, which means it is user-friendly as well as offer simple and consistent high-level APIs reduce the cognitive load for the users as much as possible.

3.2.1. Project Workflow

To better develop and implement our models, we decided to use some common deep learning software and platforms. After we finished collecting the raw datasets, we stored all the videos in the Google Cloud for preprocessing and transforming the dataset; Google Colab was utilized since it enables all team members to share and work codes together. After data preprocessing and data transformation, we split the training, validation, and testing datasets using Jupyter Notebook, since each person had their own model and prepared datasets; Jupyter Notebook is more suited for individual tasks.

Since our video dataset generated thousands of video frames, it required us to have an effective environment when training our model. We kept using Google Colab to train our model after we uploaded the prepared dataset, but we considered purchasing extra hardware acceleration in order to improve the training speed. When we finished the training and testing process, we developed a platform to auto-generate crime types once we uploaded the videos.

3.2.2. System Architecture

As this project is video-based crime detection, the user who is registered to monitor the surveillance videos with our models will help to detect any criminal activities happening around. We initially converted those input videos to frames according to the respective input size for the models proposed above. We then labeled those frames using different annotation tools like Roboflow and Lableme. We then perform data preprocessing methods like resizing, data augmentation, applying transformations, feature scaling, and data split on those annotated images. To facilitate analysis and utilization, we preprocessed data. Disposing of redundant or inconsistent information improved the reliability of the model. We used feature extraction to cut down on the number of necessary components without sacrificing accuracy. We utilized Google Drive as our database [23] to store all kinds of data across this process. We developed an integrated system for arson, burglary, and vandalism detection.

- **Preprocess Module:** When an input video is passed to the system, it generates video frames and processes the frames with resizing and data augmentation techniques by each model.
- **Object detection:** Each model processes these frames and may or may not provide detected objects for a specific crime type. This function is supported by three models: YOLOv5 provides arson objects, YOLOv7 provides burglary objects, and YOLOv6 provides vandalism objects.
- **Classifiers:** The detected objects are then classified based on the threshold to see whether a crime exists or not. Based on the threshold scores, the decision system will decide which crime the objects belong to.
- **Display summary:** Decision system crime type is displayed to the registered user along with bounding box, predicted scores, crime type objects, and classified crime. Additionally, a notification is sent as soon as the decision system classifies crime.

Table 5 lists a summary of all sections including crime types, sources, total dataset, models, and performances. We can see that YOLOv5 outperforms all other models; however, in this project, we cannot compare the results because the models were trained on different datasets for each crime type.

Table 5. Summary Table of All Sections.

Crime Type	Sources	Total Dataset	Models	Average Precision	Average Recall	mAP@50
Arson	UCF, Youtube, Storyblocks	8050	YOLOv5	0.95	0.86	0.80
Burglary	UCF, Youtube, Storyblocks	9100	YOLOv7	0.89	0.86	0.87
Vandalism	UCF, Youtube, Storyblocks	9050	YOLOv6	0.88	0.79	0.86

3.3. Model Comparison and Justification

Model Comparison

For our project, we proposed three modified models, an Improved YOLOv5 model for arson, improved Faster RCNN, YOLOv7, for burglary, and an improved SSD MobileNet [24], YOLOv6, for vandalism. Table 6 shows mainly the comparison of ML/DL models and their corresponding accuracy.

Table 6. Model comparison.

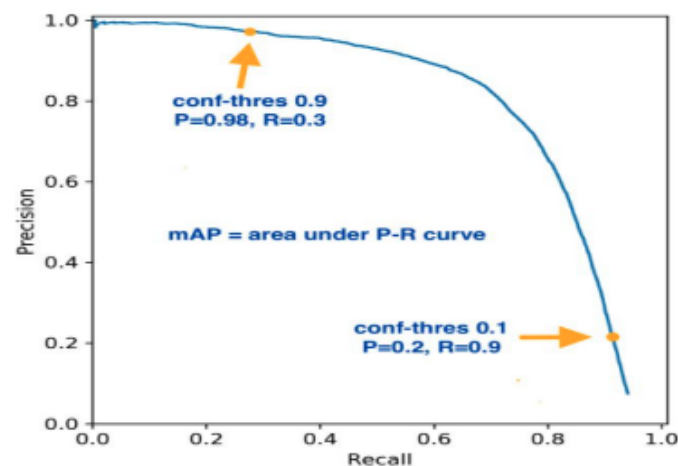
Crime Type	Model Object Detection	Existing Model	mAP@50
Arson	Improved YOLOv5	No	80%
Burglary	Improved Faster RCNN	No	61%
	Improved YOLOv7	No	87%
Vandalism	Improved SSD MobileNet	No	67%
	Improved YOLOv6	No	86%

Since Faster RCNN and SSD MobileNet models [25] provide an mAP of less than 70%, we considered YOLOv7 for burglary and YOLOv6 for vandalism.

3.4. Model Evaluation Methods

To evaluate the model performance for all the crime detection and classification models, we used the following metrics.

- Intersection over union (IOU): The metric provides true objects detected correctly and true objects detected incorrectly with respect to the ground-truth of labeled regions and detected regions.
- Mean Average Precision (mAP): To evaluate the valid set and test test, we will use mAP. As shown in Figure 14, it provides the area under precision and recall curve that helps to understand the burglary, arson, and vandalism objects detected correctly and incorrectly.

**Figure 14.** Mean average precision (mAP).

- We evaluated using the training loss and validation loss to see whether the model trade-off for bias and variance was acceptable.
- mAP accuracy was also utilized in evaluating the training, and validation sets.
- We evaluated the speed at which our object detection model processed a video and produced the correct output, measured in frames per second (FPS).
- For machine learning classification issues with multi-class output, we evaluated them using a confusion matrix.

3.5. Model Validation and Evaluation Results

3.5.1. Improved YOLOv5

The input frames were scaled to $224 \times 224 \times 3$ as the input of the improved YOLOv5 model. We split the data into training, validation and test datasets with 80%, 15%, and 5% proportions. To test the trained model, we used 6440 frames, which provided about 80% accuracy. Consequently, we were able to validate 80% of the data. We then ran a test set of 1208 unseen photos through the model with an epoch of 250 and a batch

size of 64, and achieved an precision of about 95% with a model summary of 270 layers, 7,027,720 parameters, and 16.0 GFLOPs. If we look at Figure 15, we can see that the validation loss reduces rapidly during the first 50 epochs of model training, the loss decreases slowly before reaching a plateau around 100 epochs of training. The precision for the training dataset was 95%, with a recall of 86%; for the validation dataset, the mean average precision 90% and 95%, respectively.

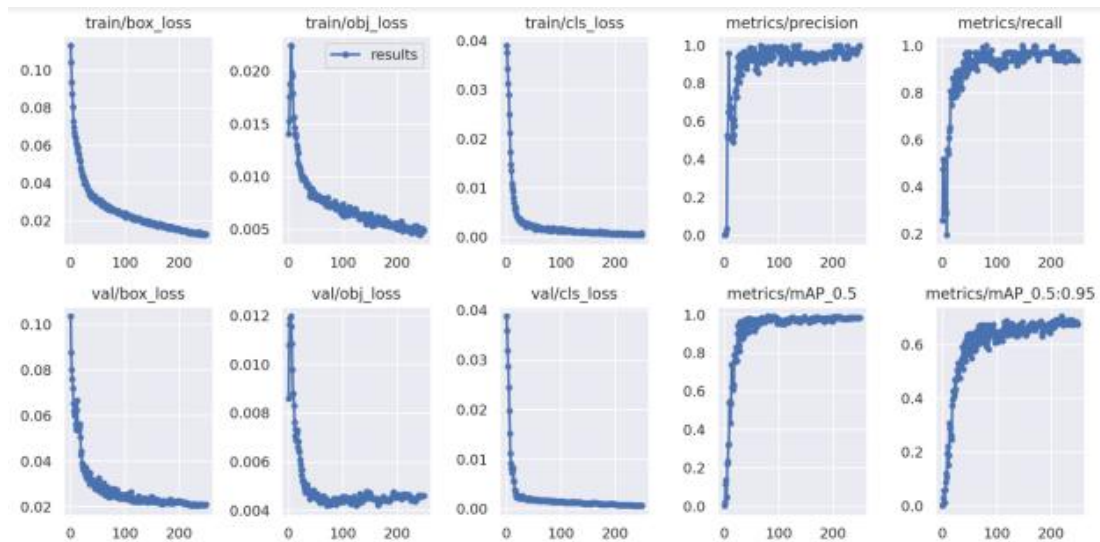


Figure 15. Evaluation results for Improved YOLOv5.

3.5.2. Improved YOLOv7

For YOLOv7, all the input images were resized to $224 \times 224 \times 3$. However, this model could be trained on any image size. We divided our dataset into training, testing, and validation with a proportion of 80%, 5%, and 15%, respectively. To test the trained model, we used 7280 images that provided 89% precision. To test our dataset, we use 455 images through the model with 250 epochs and a batch size of 16. This gave us around 87% of the mAP. Figure 16 shows that when we train the model for the first 20 epochs, validation loss is coordinated with training loss. Then, for the next 20 epochs, the training and validation loss decrease. We can say that the model improved, Although the model has reduced loss, the precision shows 89% and recall shows 86% for the training set and test set. The mean average precision shows 87% and increases efficiency and accuracy, as we have introduced more datasets by collecting more videos or through data augmentation, changing batch size, learning rate and experimenting with different optimizers, and freezing the last three layers of the convolution layers.

3.5.3. Improved YOLOv6

All of the input pictures were scaled up to 224 by 224 by 3 for YOLOv6. Despite this, the model may be trained using images of arbitrary dimensions. We split our dataset up into three sections: a training set, a test set, and a validation set, with 80%, 5%, and 15% proportion, correspondingly. A total of 7140 pictures were utilized for testing the trained model, yielding an accuracy of 88%. We ran a total of 386 photos through our model with a total of 550 epochs and a batch size of 32 for our dataset validation. In total, this covered almost 86% of the mAP! During the first 30 epochs of model training, as shown in Figure 17, validation loss is synchronized with training loss. The train and the validation loss then go down for the following 50 epochs. Overall, the model does get better. While the model does demonstrate reduced loss, it nevertheless has low precision (88%) and recall (79%). As we have presented more datasets by collecting further videos either through data augmentation, altering batch size, learning rate, experimenting with different optimization techniques, and freezing the very last three layers of the convolutional layers, the mean

average precision has increased to 86%, indicating improved efficiency and accuracy. With more data and more time periods, the model has shown substantial improvement.

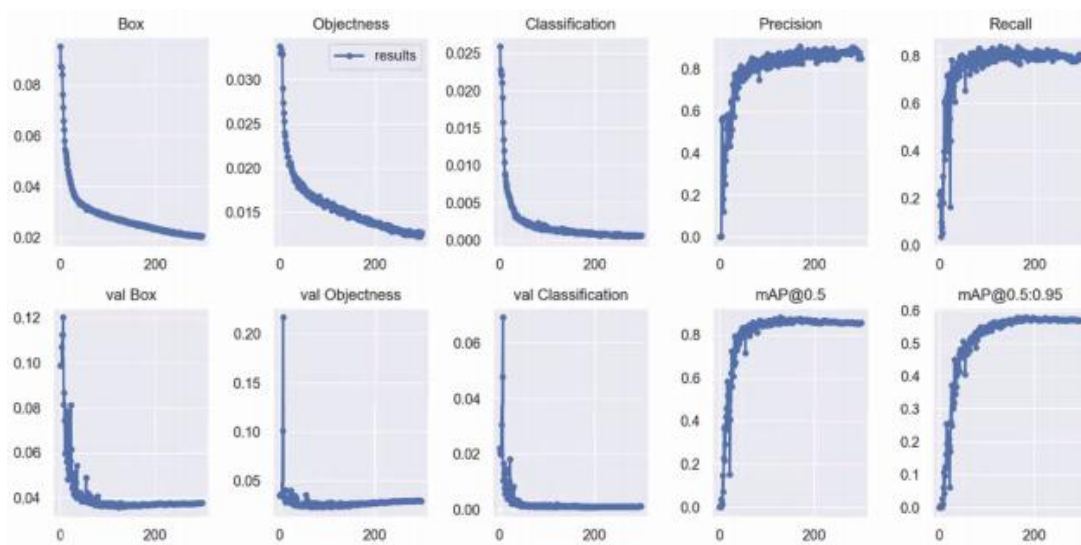


Figure 16. Evaluation results for Improved YOLOv7.

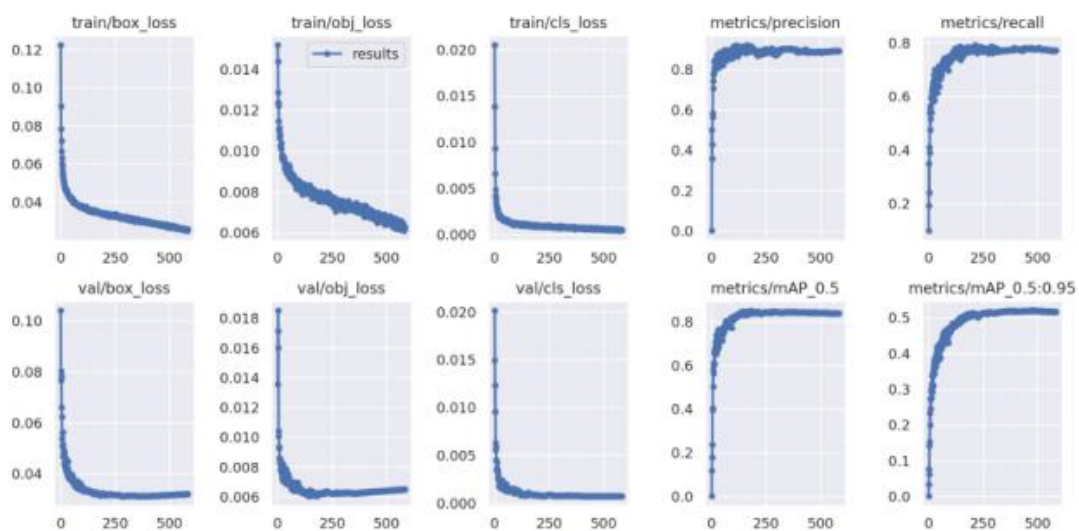


Figure 17. Evaluation results for Improved YOLOv6.

3.5.4. Model Comparison

Table 7 compares the accuracy of different models for detecting various crime types. As shown in the table, for Arson detection, the YOLOv5 model achieved an average precision of 0.95 and an average recall of 0.86 on a dataset of 8050 samples, with an mAP@50 FPS of 0.80 and a processing speed of 15 ms. For Burglary, the YOLOv7 model demonstrated excellent performance, achieving an average precision of 0.89 and an average recall of 0.86 on a dataset of 9100 samples, with a faster processing speed of 12 ms and an mAP@50 FPS of 0.87. Lastly, for Vandalism, the YOLOv6 model attained an average precision of 0.88 and an average recall of 0.79 on a dataset of 9050 samples, maintaining a competitive mAP@50 FPS of 0.86 and a processing speed of 15 ms.

Table 7. Model accuracy comparison.

Crime Type	Model	Dataset Size	Average Precision	Average Recall	mAP@50	FPS (ms)
Arson	YOLOv5	8050	0.95	0.86	0.80	15
Burglary	YOLOv7	9100	0.89	0.86	0.87	12
Vandalism	YOLOv6	9050	0.88	0.79	0.86	15

4. Data Analytics and Intelligent System

4.1. System Requirements Analysis

System Boundary and Use Cases

Figure 18 shows the potential use cases of this project could be in HOA communities, apartment buildings, public parking, or street parking. The users of this system would be home owners, apartment owners/renters, and parking lot owners. As illustrated in the figure, the possible scenarios are applicable when a surveillance [26] camera installed inside or outside a house, building, or public parking captures the behaviors happening around the neighborhood. Arson: If a person is setting fire around a house in an HOA community or an apartment building to steal things or other suspicious activities [27], the system detects such activities and classifies them as arson and sends an email to the registered user. Similarly, if a person sets fire to a vehicle in a public parking space, street parking space, apartment parking lot, or community parking, the system sends identified objects and the classified crime to the registered user. Burglary: If a person is breaking into a house in an HOA community or apartment building to steal things or other suspicious activities, the system detects such activities and classifies them as a burglary and sends an email to the registered user. Similarly, if a person breaks into a vehicle in a public parking space, street parking space, apartment parking lot, or community parking to steal something inside a car, the system sends identified objects and classified crimes to the registered user. Vandalism: If a person is breaking/destroying any public property in HOA community or parking lots or any suspicious activity related to it, the system detects such activities and classifies them as vandalism and sends an alert message to the registered user, which includes identified objects and crime type.



Figure 18. System boundary and possible use cases.

The aim of the project is to develop and deliver a robust automated intelligence platform that detects and classifies specific crime types (vandalism, burglary, and arson) within HOA communities, public parking spaces, and apartment buildings, from camera-

based videos and surveillance videos [28]. The system uses diversified video datasets such as day and night, different camera angles, etc.

Data analytics for the system are prepared in four aspects: elicitation, analysis, specification, and validation and verification. In the elicitation process, specific crime-related characteristics are processed and trained by deep learning models. Further, the video frames captured by surveillance videos are used to generate particular crime activities, and then, using the decision table, the crime is classified.

The specification requirements involve the data requirements. Training data are collected and processed in different shapes, sizes, formats, and resolutions. The input data are passed to the surveillance camera and processed by trained ML models to provide detected objects for different crime types.

Table 8 outlines the system data requirements for the project, focusing on detecting and classifying different crime types in both daytime and nighttime scenarios. The key requirements include high spatial resolution videos for each crime type, including arson, burglary, and vandalism.

Table 8. System data requirements for project.

Tasks	Data Requirements	Labeling Tool	Label Export Format
Detect objects in arson and classify the crime type (during daytime and night time)	High spatial resolution videos	Roboflow	.txt
Detect objects in burglary and classify the crime type (during daytime and nighttime)	High spatial resolution videos	Roboflow	.txt
Detect objects in vandalism and classify the crime type (during daytime and nighttime)	High spatial resolution videos	Roboflow	.txt

To ensure whether the training data match the real data, validation requirements should be performed often. To maintain the optimal performance of the models, we need to monitor and analyze the runtime data. Hence, the ML systems should be retrained often to adjust to the real data.

4.2. System Design

4.2.1. System Architecture and Infrastructure

Figure 19 depicts the front-end and back-end architecture of the system used to detect and categorize criminal activity. Initially, users are registered in a communication tool 'Twilio', which they use to receive text messages related to crime. Furthermore, we use the 'Gradio' in built applications in Python to design the user interface. In one possible situation, Google Drive is used to save footage captured during an individual's attempt at committing a criminal act, such as arson, theft, or property destruction. All the work for the preprocessing, detection, and classification models [8] is done in the drive itself using Google Colab. Data augmentation, such as scaling, flipping, rotations, etc., and frame generation falls under preprocessing. All the extracted frames are then uploaded to Roboflow to perform annotations on each frame. We then resize the frames to their respective dimensions in accordance with the models at the resize stage. Our object detection system makes use of YOLOv5, YOLOv7, and YOLOv6. With the aid of scores and bounding boxes, these models can pinpoint the specific items that play a role in illegal actions. Those predictions [4] are performed on video data to check the detection frame by frame. Following this, the data are sent to the OpenCV decision model, where a logic specific to each crime is entered to aid in the classification process (here, arson, burglary, or vandalism). The logic behind it is that it checks for specific objects in an order, as shown in Figure 19. If the predicted objects include fire or (person + fire) or (car + person + fire),

then it displays as arson on the video. If not, it checks for the burglary objects that include break_in or (person + break_in) or (person + break_in + stealing) and, if present, they display as a burglary in the video. Otherwise, it checks for the vandalism objects that include destroying or (person + destroying) or (person + scratching + car + destroying); then, it is displayed as vandalism in the video itself. In the scenario where none of the above combination objects are detected in a video, then it displays as normal in the video. After the classification of the crime type, if there is a presence of a crime in the video, an alert message is sent to the registered user. In the case of an unregistered user, a warning message is also triggered within the UI. Users are notified via alert system if any of these criminal acts have occurred. The alert message includes specifics on the objects and the sort of crime that have been spotted.

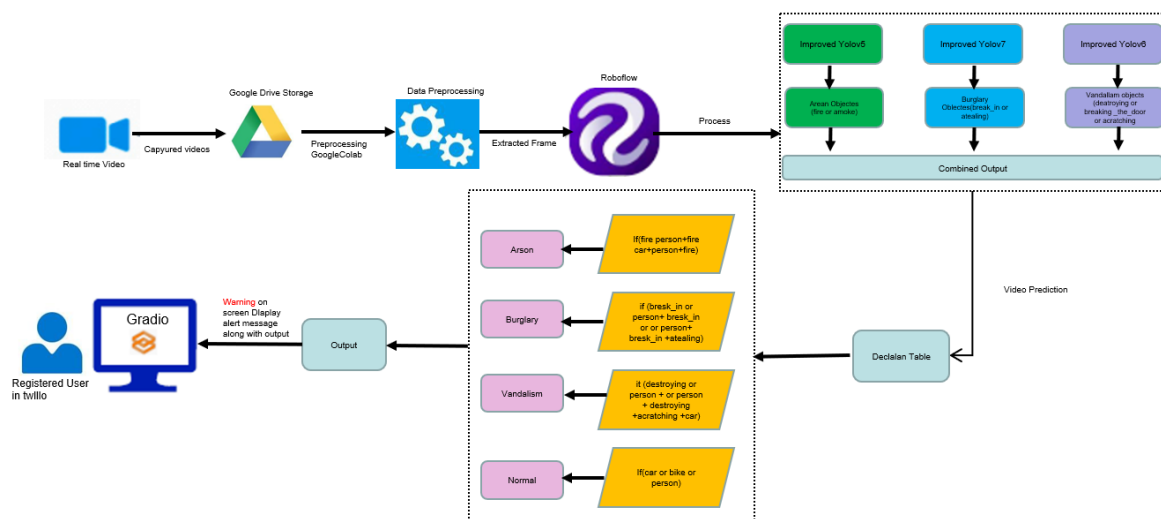


Figure 19. Front-end and back-end system architecture.

4.2.2. System Data Management Solution

First of all, all the users need to register for the system, and upload the relevant video from a local server or cloud server on the user interface website using the Gradio package. When the user finishes uploading the video, it is transmitted to the crime object detection system, which captures the user's videos. Each input video is sent to the integrated model with all the information trained by the object detection system and the saved output, such as classes and positions; all the data are saved and updated in real time. Next, the video detection outputs are sent to classification; after finishing the whole process, all the outputs are displayed on the website to make predictions with the matched videos. When it comes to crime videos, the system generates alerts and sends alert messages to the current registered users; if there is no crime involved in the video, only the output to the user interface with no alert generation is displayed.

4.3. System Development

4.3.1. AI and Machine Learning Model Development

In order to detect and classify the recognition problems [29], we used a development learning system with the support of various deep learning models. Then, we looked for approximated functions as the learning problem. To avoid errors in the learning process, several factors were considered:

- The framing and observations used to teach the model.
- Preparation of the training, validation, and test data.
- Teaching the algorithm to fit the model on the training data.
- Evaluating the models' performance using different metrics.

The system was designed to detect crime objects and also provide crime classification. Hence, we chose to frame the learning based on several crime activities and then classify them in the end. The observations used to train the models are crime and non-crime videos with different camera angles. We collected data from several websites for each of our crimes. We proposed three different improved deep learning models as the predictive models and their performance was measured with mAP accuracy, misclassification rate, precision, recall curves, predicted scores, validation, and test results. To do the classification, initially, the proposed deep learning models were YOLOv5, Faster RCNN, and SSD MobileNet. All models were combined into our system after being trained and tested independently to provide the crime classification.

Improved YOLOv5 was used to train on the arson dataset in order to detect crime behaviors relevant to arson and target objects related to it. YOLOv5 is mainly composed of four parts: input, backbone, neck and head. The backbone is the convolutional neural network [30] that aggregates and forms image features on fine-grained scales of different images, the neck is a series of network layers that mix and combine image features and pass image features to the prediction layer, and the head is used to predict image features, generate bounding boxes, and predict target object categories. In the Improved YOLOv5 model, we froze the conv-layers from the backbone part to accelerate training and not change the weight for transfer learning, we also changed the batch size to reduce the computation cost, and added dropouts to avoid the overfitting problem.

Improved Faster RCNN ResNet101 was trained on the burglary dataset to detect crime activities and objects related to burglary. ResNet101 has $7 \times 7 \times$ sixty-four convolution layers, 33 building blocks, and three layers that make it 99 layers and add up to $1 + 99 + 1 = 101$ layers. Furthermore, ResNet101 uses an FPN network to extract features that solve multi-scale problems. The Faster RCNN has RPN and ROI networks; RPN fails to recognize smaller objects with different angles. In the improved Faster RCNN model, for better detection, images are cropped and enlarged to learn the features. Additionally, after pre-training the model, we froze the layers of ResNet101 to reduce the significant training time and computation cost. We also modified the batch size to eight to avoid out-of-memory issues, increased the number of steps for better detection performance, and added more training data to avoid overfitting issues. Then, for the output layer, the softmax function is set to the five defined burglary classes to detect burglary crime-related objects and activities.

YOLOv7 was trained on the burglary dataset to detect crime-related behaviors and activities. The model was enhanced on YOLOv4, Scaled YOLOv4, and YoloR. It has five main components: inputs, backbone, Feature Pyramid Network, neck, and head. Once the input is given, the backbone aggregates the convolution layers and passes information to the FPN to recognize small objects. The extracted features are then sent to the neck to predict the features, whereas the head predicts the objects and also provides confidence based on regression of these predicted features. To improve the performance of the model, we froze the last three layers to avoid overfitting, changed the batch size to 16 to avoid out-of-memory issues, set the output layer and the softmax function to the five defined burglary classes to detect the burglary crime-related objects and activities, and increased number of epochs for better average precision, and mAP.

SSD MobileNet is a one-shot solution capable of recognizing several objects in a single photograph. If you split depth-wise and point-wise convolutions apart, MobileNet has 28 layers of neural networks. Classification and detection tasks benefit greatly from the base network's elevated properties. An additional completely linked layer is useful for classification in these networks, followed by a softmax layer. The SSD detection networks approach uses a feed-forward convolutional network to provide a collection of bounding boxes of a predetermined size, together with scores indicating the likelihood that instances of a given object class lie within the boxes. We refined the model by freezing all layers except the output layer in light of the training data's intended purpose, the desired output, and the classification challenge for the mask position. The model was trained with a

learning rate of 0.15. In order to prevent overfitting, the improved SSD mobilenetv2 adds a dropout layer after the classifier's softmax layer, which converts the final layer's output to probabilities. For the model to be able to recognize and classify images in a wide range of lighting situations and resolutions, it must be combined with additional data preparation methods [31].

YOLOv6 was trained on the vandalism dataset to detect the crime and its related objects. This model was introduced to address the flaws present in YOLOv5. YOLOv6 provides a range of models for a wide range of industrial use cases, from N to T to S to M to L, with varying architectures depending on model size to achieve an optimal balance between speed and accuracy. Additional performance enhancements are introduced in the form of self-distillation and additional training epochs, both of which are part of the bag-of-freebies methodology. Our goal in adopting QAT for widespread use in industry is to achieve maximum performance through the use of channel-wise distillation and graph optimization. On the COCO dataset, YOLOv6-N achieved 35.9% AP at 1234 FPS on T4. On T4, YOLOv6-S hit 43.5% AP at 495 FPS, and the quantized version hit 43.3% AP at 869 FPS. In addition to its superior performance, YOLOv6-T/M/L also demonstrated superior accuracy compared to other detectors while maintaining a comparable inference speed. There have been some structural changes made in YOLOv6 compared to earlier versions. An mAP representation of the input is provided by the structure's backbone.

Moreover, it has a neck that separates off intricate details from the input. The final result is calculated by the brain. The EFFICIENTrep backbone is used in YOLOv6, and it can take advantage of specialized computer [32] capabilities like a GPU. The Rep-PAN Neck used in this edition is both quicker and more precise. Decoupling the head adds a layer between the network and the terminal node, which boosts the network's efficiency. These changes to the underlying structure are what make YOLOv6 so much faster than its predecessors and give it other benefits.

Integrated model: To build the integrated model, we used the OpenCV library. Initially, we saved the trained individual models (arson, burglary, and vandalism) as .pt file weights that detect the target objects for each crime type (YOLOv5, YOLOv7, and YOLOv6). When we pass a video, each frame goes through all three model predictions and detects crime-related objects; in parallel, the crime is classified based on the time for the detected objects. To classify the crime, one frame per second is used with a set threshold > 0.5 and a count is considered to detect objects. If crime-related objects are detected repeatedly for about 5–7 s, then, for a specific crime, it is classified and displayed on the video. Below are the scenarios that were tested using the integrated model.

Scenario 1 (burglary): A video is passed and each frame goes through each crime model; one of the models predicts either (break_in) or (person and break_in) or (person and break_in and stealing) repeatedly, with high thresholds for about 7 s, and the other two models do not detect anything; then, the crime will be classified as a burglary. These detected and classified crime frames are converted into a video that showcases bounding boxes, scores, labels, and classified crime type. This is stored in the specified path for further application development.

Scenario 2 (arson): A video is passed and each frame goes through each crime model; one of the models predicts either (fire) or (person and fire) or (person and fire and car) repeatedly, with high thresholds for about 3 s, and the other two models do not detect anything; then, the crime is classified as arson. These detected and classified crime frames are converted into a video that showcases bounding boxes, scores, labels, and classified crime type. This is stored in the specified path for further application development.

Scenario 3 (vandalism): A video is passed and each frame goes through each crime model; one of the models predicts either (destroying) or (person and destroying) or (person and destroying and scratching) or (person and car and scratching) repeatedly, with high thresholds for about 5 s and the other two models do not detect anything; then, the crime is classified as vandalism. These detected and classified crime frames are converted into

a video that showcases bounding boxes, scores, labels, and classified crime type. This is stored in the specified path for further application development.

Scenario 4 (normal): A video is passed and each frame goes through each crime model; the models predict non-crime objects such as person, car, bike, etc., repeatedly with high thresholds for about 7 s; then, the crime is classified as normal. These detected and classified crime frames are converted into a video that showcases bounding boxes, scores, labels, and classified crime type. This is stored in the specified path for further application development.

4.3.2. Implement Designed System

In this paper, the designed system includes suspicious target object detection, crime, or normal video classification and crime type classification. Input videos are recorded by surveillance camera and saved into the cloud, then fed into the detection process. All the detection is performed by the corresponding improved model and each model classifies and generates a binary output of crime type or normal video, based [33] on the corresponding detected object.

In terms of the functional component of the designed system, one is the object detection by improved deep learning models; another functional component is the integrated algorithm [34], which could apply to improved models in a hybrid method, as well as combined in a sequential way which concatenate to all models. If the video contains suspicious target object, first, we consider if the video belongs to arson and if its classification will be performed by Improved YOLOv5; if yes, then we generate the output as well as an alert; otherwise, we keep classifying; if the video belongs to burglary and its classification will be performed by Improved YOLOv7, then we generate the output; if not, classification will continue to vandalism, performed by Improved YOLOv6; after we process classification through all the crime types, but if we cannot define the crime, then the output as a normal video will be generated without an alert.

4.3.3. Input and Output Requirements, Supporting Systems, and Solution APIs

- **Input datasets:** The input dataset could be recorded from a surveillance camera and could be any videos that are available to detect. Once the videos are uploaded to the system which is the open platform, it will process the input information and will be saved into the system as the user enters. Both crime videos and normal videos are accepted with different scenarios, such as nighttime and daytime videos, and also it is better to upload high resolution input videos, which lead to better detection as well as getting a clear output.
- **Expected output:** As we mentioned in the previous section, the designed interactive system shows the output containing the input video along with their target objects with corresponding bounding boxes, labels, scores, and classified crime based on time. Furthermore, a message tells our user whether the input dataset is a crime video or a normal video; an alert will pop out if it belongs to a crime video and is also classified crime type.

Arson crime expected output: output contains target objects such as fire, person, car, and more within the video.

Burglary crime expected output: output contains target objects such as break in, stealing, person, and more within the video.

Vandalism crime expected output: output contains target objects such as breaking the door, scratching, destroying, and more within the video.

- **Supporting system contexts:** Surveillance camera or required videos in good resolution which could capture the crime behaviors within the videos.
- **Solution APIs:** For the better development of this project, we planned to use the Gradio's API built specifically for ML and Data Science projects; it is as close to the recognition as we could get, and it pre-configured most of the common recognition tasks. On the other hand, Gradio's API could build websites in dozens of lines for

Python and create an API simply, and also provide comprehensive deep learning packages and libraries, which are the most common uses of it. The following lists some features and functionalities of Gradio.

API [12]: Powerful Features: Free and open sources, no callbacks, build applications with simple API, no hidden state, works with multi-packages like Tensorflow, Keras, Pytorch and more common functionalities, sketch recognition, question answering, image segmentation, time series forecasting, and XGBoost with explainability [35].

Gradio enables users to use the integrated systems within applications and its powerful functionality makes detection easy; however, there are some downsides to it; since the API right now is not comprehensive, we are more likely to get stuck into the speed issue due to the application flow.

Twilio's API will also be used in our project to further generate alert messages to registered users. It enables voice, message, as well as video contact in the website and mobile apps. The following lists the example use cases of Twilio's API:

- Track and contact real estate agents as well as house buyers.
- Users can communicate with each other without revealing their private information.
- Staff can help customers via their web interface.
- Staff receives auto-alerts at the time when vending machines need maintenance.

4.4. System Supporting Environment

Based on current open source tools, we want to create a crime detection and classification system and its accompanying frameworks with integrated solution tools for this project. We use the Twilio communication tool to register the users. Twilio using application programming interfaces (APIs), previously exclusive channels such as audio, text, chat, video, and email, are now accessible to any business, facilitating the development of meaningful connections with users via their preferred channels. To represent a Twilio-built app, we can use the application instance resource. Within Twilio, an application is a collection of URLs and some other configuration files that instruct Twilio on how to handle a call or SMS message received by one of our Twilio numbers. These numbers will be required at the time of sending alert messages related to crime behavior. Model creation is handled on the back-end with the help of Keras, Tensorflow, Pytorch, and OpenCV. Google Colab is where we manage the combination of three models and the image processing of our training data. Object recognition, localization, and tracking [36] in TensorFlow are all computer vision techniques [37]. This technique not only helps us understand the image or video better by detecting items, but it also sheds light on the inner workings of the models used to do so. The primary benefit of using Pytorch for object detection is that it defines the class of objects and their positions based on the supplied data. Assigning anything to a class indicates that it belongs to a specific group of things, such as people, systems, tables, etc. All the data used for training are stored locally as well as in Google Cloud. Large datasets can be obtained and trained for complicated upgraded deep learning models with the help of Colab and Google Cloud.

5. System Evaluation and Visualization

5.1. Analysis of Model Execution and Evaluation Results

In comparison to previous mean Average Precision, we fine-tuned and improved our models' mAP. Each model was evaluated using different metrics such as mAP, average precision, average recall, fps, train and validation loss curves, and confusion metrics. The YOLOv5 model showed high improvement from 50% to 80% after training for 250 epochs. YOLOv7 [38] also had significant improvement from 61% to 87%, whereas YOLOv6 after training for 550 epochs had high improvement of 86% when compared to the mAP produced by SSD, which was 67%, as shown in Table 9. All of the models performed better when high resolution annotations were added. In Figure 20, for burglary, we can see break_in, stealing, person, and car are classified with high thresholds, however, bikes

and break_in are misclassified with high thresholds. For arson, even though the model had excellent performance, it detected fire with a high threshold for light. For vandalism, cars were misclassified. In order to improve these results, the last 10 layers were frozen in the YOLOv5 model, performing better compared to original models. YOLOv7 and YOLOv6 improved after changing the learning rate, batch size, number of steps, and additional data for frozen convolution base.

Table 9. Model improvement comparison.

Crime Type	Model	Previous mAP@50	Current mAP@50
Arson	YOLOv5	0.50	0.80
Burglary	YOLOv7	0.61	0.87
Vandalism	YOLOv6	0.67	0.86



Figure 20. Sample output from our integrated system.

5.2. Achievements and Constraints

Our goal for this project was to build a multi-crime type detection recognition website and, in order to achieve our goal, we need to complete the following tasks:

- Implement model for each crime type and integrate in one hybrid model.
- Present all crime type results for the user interface.
- Improve crime recognition system using OpenCV.
- Create an interactive user interface platform by using Gradio API.

Regarding the deployment of the model we split it into two parts: first we used our self-collected dataset to train the model for each crime type in order to complete the object detection using Tensorflow/PyTorch; the input image was fed into the deep learning model and generated outputs with bounding boxes as well as their class name and their coordinates. The second part was the classification: we fed the output of the first step using OpenCV to classify crime type by concatenating each time step. The advantage is that the display output easily showed what crime type the input video belongs to; however, there are some drawbacks which cannot be ignored. With the large amount of computation cost and due to the limitation of our computer power, generating output by using combined architecture becomes much more time consuming. On the other hand, we could not guarantee the quality for every single video, so there might be some detections which were not correct, and it is obvious that there should be some improvements in the video resolutions.

After we trained, fine-tuned, and saved the individual model to the best.pt file, we used OpenCV packages to further deploy our crime system. We integrated our model in a hybrid way with burglary crime detection, arson crime detection, vandalism crime detection, and normal detection; detection was deployed in parallel and the integrated model was combined with the crime detection recognition system. The combined model was fed with different image sizes as input, 224×224 for the Improved YOLOv5 model for arson detection, 640×640 for the Improved YOLOv7 model for burglary detection [39], and 640×640 for the Improved YOLOv6 model for vandalism detection. The integrated model will generate a combined output displayed with crime type results (arson, burglary, vandalism, and normal), the result will be based on time, whether the video contains the potential criminal, along with their crime behaviors for certain seconds.

The system could also detect the target objects, which are related to the crime type and will be able to show the crime result on the left top corner, as shown in Figure 20; if the video is a crime video, then the crime type is going to be represented using the color yellow and also show the bounding boxes for the target objects; if the video is a normal video and contains no crime behavior, then the output will display the word normal on the left top corner, in yellow as well.

To better visualize the integrated system, we are going to connect Gradio API to our detection system in order to create an interactive website, in such a way that our users could use the system by accessing the URL link online and look for the results by themselves. Additionally, to improve the performance of our crime detection system, Gradio API is going to be implemented, since we can easily upload and display video in it. However, there are some constraints that make the deployment of our integrated system difficult. We do not have a full toolset for Gradio, due to the fewer tools, so we may need to build more on our own or search out more extensions/libraries from external sources.

When we are not able to detect any crime behaviors within the video we just simply define the video as normal video. The following Figure 20c,d show the outputs of both daytime and nighttime normal video, the result is correctly displayed on the top left corner using the color yellow and no crime behavior involved within the video.

In this detection and classification, we are considering both daytime and nighttime video datasets. For crime type detection, we used decision flow by OpenCV to get predictions in the video along with the crime type. The following Figure 20e–g show the output of daytime video; the crime type is displayed on the left top corner as well as the located target objects, represented by the bounding boxes with their class names; the model successfully detects for crime video and displays the crime type results.

We not only detect the crimes happening during the daytime but also nighttime. The following Figure 20i,j show the output of a nighttime video; the crime type is displayed on the left top corner as well as the located target objects, represented by the bounding boxes with their class names; the model successfully detects for crime video and displays the crime type results.

5.3. System Quality Evaluation of Model Functions and Performance

Machine learning, at its core, is the meeting of statistical methods with computational analysis. Machine learning relies on algorithms and models, which are essentially boosted statistical estimations. Evaluating models on test data is called model evaluation. The test data is made up of information the model has never seen before. Assigning a probability to each classification, the model's accuracy can be assessed. Every model and category has a unique probability [40]. Every model's performance is measured in terms of its runtime in seconds. The runtime duration varies widely between the devices and GPU. For our models we have used Google Colab PRO. In Table 10; the average run time and device specifications are compared.

Table 10. Run time performance comparison.

Features	Device Specifications	Average Run Time (s)	Google Collab Pro	Average Run Time (s)
Arson object detection	Macbook Pro Processor: 2.6 GHz 6-Core Intel Core i8 Memory: 16 GB 2667 MHz	1.93	Google Cloud Pro Platform Deep learning VM with GPU (NVIDIA Tesla K80, P100, T4) 32 GB RAM	1.23
Burglary object detection	Macbook Pro Processor: 2.6 GHz 6-Core Intel Core i7 Memory: 16 GB 2667 MHz	2.14	Google Cloud Pro Platform Deep learning VM with GPU (NVIDIA Tesla K80, P100, T4) 32 GB RAM	1.12
Vandalism object detection	HP Windows 10 (64bit OS) Processor: Intel® Core™ i7-10510U Memory: 16 GB 2300 MHz	2.31	Google Cloud Pro Platform Deep learning VM with GPU (NVIDIA Tesla K80, P100, T4) 32 GB RAM	1.26
Integrated Model	Macbook Pro Processor: 2.6 GHz 6-Core Intel Core i7 Memory: 16 GB 2667 MHz	2.45	Google Cloud Pro Platform Deep learning VM with GPU (NVIDIA Tesla K80, P100, T4) 32 GB RAM	2.0

6. Conclusions

6.1. Summary

This paper designed an automatic identification [41] of crime types based on camera or surveillance video in the absence of an unattended person, and alerts registered users about crimes such as arson, burglary, and vandalism through SMS service. To detect the object of the crime in the video, we trained five different machine learning models: Improved YOLOv5 for arson, Faster RCNN and YOLOv7 for burglary, and SSD MobileNet and YOLOv6 for vandalism. Other than improved models, we have innovated by building ensemble models of all three crime types. We have utilized the OpenCV library to read, process, detect objects [42], classify crime, and write frames to display video. This article completes the detection and classification [43] of AI-based smart crime cameras that help in many crime (arson, burglary, and vandalism) situations: HOA neighborhoods, parking lots, public parking lots, street parking, construction, and apartments, and generate alerts for homeowners via SMS services [44].

6.2. Prospect and Future Work

Benefits of the solution: The main aim of the project is to provide security to society without human involvement and make affordable surveillance cameras to detect and classify crimes. The advantages of this project are listed below:

- Provide more security [45] for arson, burglary, and vandalism crimes in the public and private space such as HOA communities, apartment buildings, street or public parkings, and for individual homeowners.
- Easy to use website where a user can do simple free registration by creating a Twilio account to get an SMS alert as soon as the video classifies any crime.

- Since we are using YOLO models, it is easy to use the OpenCV library to detect our customized objects in videos.
- Different object detection models [46] helped us to understand how other models performed, which benefited us to navigate for new YOLO models compared to Faster RCNN and SSD MobileNet and assisted in saving training time and computation cost as the other two models had huge parameters.
- To get the detections and classification on any given video, we have assembled all three models rather than using a single model for each crime type.
Shortcoming of the solution:
 - To get alerts for crime types, users have to register through Twilio accounts. However, users can use a free trial for up to 30 days. In order to have an active account, users have to pay monthly or annually.
 - As of now, we have trained models for arson, burglary, and vandalism crime types. Hence, our models will not be used to detect other types of crimes.
 - Our class annotations are restricted due to the limited data accessibility, high resolution videos, and the given time constraint of the semester. We can annotate more classes of all categories and increase the datasets in future for further distinctions of classes.

Future improvements to the proposed project can be made by incorporating additional types of crime records and using more sophisticated models with good resolution videos. For practical purposes, the crime dataset might be enlarged. In addition, sending frames alongside the alert messages will be more effective and useful for the people in authority to take action right away. To make the monitoring of surveillance systems more complete, further elements like person tracking can be implemented. Currently we use OpenCV for the classification of crimes. Instead, as an extension to the project, we can try using the DarkNet framework, LSTM [47], for further analysis.

Author Contributions: Methodology, J.S. and P.B.; Software, A.S. and B.Z.; Validation, B.Z. and H.Y.; Investigation, B.Z.; Data curation, J.S. and B.Z.; Writing—original draft, J.G.; Writing—review and editing, Y.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China under Grant 62006169 and supported by the Shanxi Province Research Foundation for Base Research, China (Grant No.202303021221002).

Data Availability Statement: The datasets generated and/or analyzed during the current study are not publicly available because the data were obtained from private communities with a privacy agreement but are available from the corresponding author.

Conflicts of Interest: The authors declared that they have no conflicts of interest to this work. We declare that we do not have any commercial or associative interest that represents a conflict of interest in connection with the work submitted.

References

1. Norkobil Saydirasulovich, S.; Abdusalomov, A.; Jamil, M.K.; Nasimov, R.; Kozhamzharova, D.; Cho, Y.I. A YOLOv6-Based Improved Fire Detection Approach for Smart City Environments. *Sensors* **2023**, *23*, 3161. [CrossRef] [PubMed]
2. Navalgund, U.V.; Priyadarshini, K. Crime intention detection system using deep learning. In Proceedings of the 2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICSDDET), Kottayam, India, 21–22 December 2018; pp. 1–6.
3. Ali, L.; Alnajjar, F.; Jassmi, H.A.; Gocho, M.; Khan, W.; Serhani, M.A. Performance evaluation of deep CNN-based crack detection and localization techniques for concrete structures. *Sensors* **2021**, *21*, 1688. [CrossRef] [PubMed]
4. Shah, N.; Bhagat, N.; Shah, M. Crime forecasting: A machine learning and computer vision approach to crime prediction and prevention. *Vis. Comput. Ind. Biomed. Art* **2021**, *4*, 9. [CrossRef] [PubMed]
5. Chackravarthy, S.; Schmitt, S.; Yang, L. Intelligent crime anomaly detection in smart cities using deep learning. In Proceedings of the 2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC), Philadelphia, PA, USA, 18–20 October 2018; pp. 399–404.
6. Jonathan, H. mAP (Mean Average Precision) for Object Detection. Available online: <https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173> (accessed on 25 February 2024).

7. Padilla, R.; Netto, S.L.; Da Silva, E.A. A survey on performance metrics for object-detection algorithms. In Proceedings of the 2020 International Conference on Systems, Signals and Image Processing (IWSSIP), Niterói, Brazil, 1–3 July 2020; pp. 237–242.
8. Maqsood, R.; Bajwa, U.I.; Saleem, G.; Raza, R.H.; Anwar, M.W. Anomaly recognition from surveillance videos using 3D convolution neural network. *Multimed. Tools Appl.* **2021**, *80*, 18693–18716. [[CrossRef](#)]
9. Yuan, C.; Zhang, J. Violation detection of live video based on deep learning. *Sci. Program.* **2020**, *2020*, 1895341. [[CrossRef](#)]
10. Zamri, N.; Tahir, N.; Ali, M.; Ashar, N.; Al-misreb, A. Mini-review of street crime prediction and classification methods. *J. Kejuruter* **2021**, *33*, 391. [[CrossRef](#)] [[PubMed](#)]
11. Mohandas, R.; Bhattacharya, M.; Penica, M.; Van Camp, K.; Hayes, M.J. TensorFlow Enabled Deep Learning Model Optimization for enhanced Realtime Person Detection using Raspberry Pi operating at the Edge. In Proceedings of the AICS, Dublin, Ireland, 7–8 December 2020; pp. 157–168.
12. Yang, F.; Zhang, X.; Liu, B. Video object tracking based on yolov7 and deepsort. *arXiv* **2022**, arXiv:2207.12202.
13. Sultani, W.; Chen, C.; Shah, M. Real-world anomaly detection in surveillance videos. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 6479–6488.
14. Abraham, M.; Suryawanshi, N.; Joseph, N.; Hadsul, D. Future Predicting Intelligent Camera Security System. In Proceedings of the 2021 International Conference on Innovative Trends in Information Technology (ICITIIT), Kottayam, India, 11–12 February 2021; pp. 1–6.
15. Nyajowi, T.; Oyie, N.; Ahuna, M. CNN Real-Time Detection of Vandalism Using a Hybrid-LSTM Deep Learning Neural Networks. In Proceedings of the 2021 IEEE AFRICON, Arusha, Tanzania, 13–15 September 2021; pp. 1–6.
16. Lee, J.; Shin, S.J. A Study of Video-Based Abnormal Behavior Recognition Model Using Deep Learning. *Int. J. Adv. Smart Converg.* **2020**, *9*, 115–119.
17. Tulbure, A.A.; Tulbure, A.A.; Dulf, E.H. A review on modern defect detection models using DCNNs–Deep convolutional neural networks. *J. Adv. Res.* **2022**, *35*, 33–48. [[CrossRef](#)]
18. Abid, A.; Abdalla, A.; Abid, A.; Khan, D.; Alfozan, A.; Zou, J. Gradio: Hassle-free sharing and testing of ml models in the wild. *arXiv* **2019**, arXiv:1906.02569.
19. Sylvester, R.; Greenidge, W.I. Digital storytelling: Extending the potential for struggling writers. *Read. Teach.* **2009**, *63*, 284–295. [[CrossRef](#)]
20. Li, Y.; Zhu, D.; Fan, H. An Improved Faster RCNN Marine Fish Classification Identification Algorithm. In Proceedings of the 2021 2nd International Conference on Artificial Intelligence and Computer Engineering (ICAICE), Hangzhou, China, 5–7 November 2021; pp. 126–129.
21. Khandhar, H.M.; Bhatt, C.; Le, D.N.; Sharaf, H.; Mansoor, W. Plant Disease Identification Based on Leaf Images Using Deep Learning. In *Evolution in Signal Processing and Telecommunication Networks, Proceedings of the Sixth International Conference on Microelectronics, Electromagnetics and Telecommunications (ICMEET 2021)*, Bhubaneswar, India, 27–28 August 2021; Springer: Berlin/Heidelberg, Germany, 2022; Volume 2, pp. 215–224.
22. Cheng, M.; Cai, K.; Li, M. RWF-2000: An open large scale video database for violence detection. In Proceedings of the 2020 25th International Conference on Pattern Recognition (ICPR), Milan, Italy, 10–15 January 2021; pp. 4183–4190.
23. Amrutha, C.; Jyotsna, C.; Amudha, J. Deep learning approach for suspicious activity detection from surveillance video. In Proceedings of the 2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), Bangalore, India, 5–7 March 2020; pp. 335–339.
24. Vandaele, R.; Nervo, G.A.; Gevaert, O. Topological image modification for object detection and topological image processing of skin lesions. *Sci. Rep.* **2020**, *10*, 21061. [[CrossRef](#)] [[PubMed](#)]
25. Annisaa'F, N.; Soekirno, S.; Aminah, S. Implementation of Single Shot Detector (SSD) MobileNet V2 on Disabled Patient's Hand Gesture Recognition as a Notification System. In Proceedings of the 2021 International Conference on Advanced Computer Science and Information Systems (ICACSIS), Virtual, 23–26 October 2021; pp. 1–6.
26. Iee, J.-Y.; Cho, C.-J.; Han, D.K.; Ko, H. Hierarchical spatial object detection for atm vandalism surveillance. In Proceedings of the 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Auckland, New Zealand, 27–30 November 2018; pp. 1–5.
27. Quon, J.; Bala, W.; Chen, L.; Wright, J.; Kim, L.; Han, M.; Shpanskaya, K.; Lee, E.; Tong, E.; Iv, M.; et al. Deep learning for pediatric posterior fossa tumor detection and classification: A multi-institutional study. *Am. J. Neuroradiol.* **2020**, *41*, 1718–1725. [[CrossRef](#)] [[PubMed](#)]
28. Lin, C.; Li, L.; Luo, W.; Wang, K.C.; Guo, J. Transfer learning based traffic sign recognition using inception-v3 model. *Period. Polytech. Transp. Eng.* **2019**, *47*, 242–250. [[CrossRef](#)]
29. Guo, G.; Zhang, Z. Road damage detection algorithm for improved YOLOv5. *Sci. Rep.* **2022**, *12*, 15523. [[CrossRef](#)] [[PubMed](#)]
30. Available online: <https://www.ijraset.com/research-paper/crime-activity-detection-using-ml> (accessed on 25 February 2024).
31. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
32. Liu, Y. An improved faster R-CNN for object detection. In Proceedings of the 2018 11th International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, China, 8–9 December 2018; pp. 119–123.
33. Zhang, M.; Li, L.; Wang, H.; Liu, Y.; Qin, H.; Zhao, W. Optimized compression for implementing convolutional neural networks on FPGA. *Electronics* **2019**, *8*, 295. [[CrossRef](#)]

34. Cao, C.; Wang, B.; Zhang, W.; Zeng, X.; Yan, X.; Feng, Z.; Liu, Y.; Wu, Z. An improved faster R-CNN for small object detection. *IEEE Access* **2019**, *7*, 106838–106846. [CrossRef]
35. Al-Haija, Q.A.; Smadi, M.A.; Zein-Sabatto, S. Multi-class weather classification using ResNet-18 CNN for autonomous IoT and CPS applications. In Proceedings of the 2020 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 16–18 December 2020; pp. 1586–1591.
36. VGG16—Convolutional Network for Classification and Detection. Available online: <https://neurohive.io/en/popular-networks/vgg16/> (accessed on 25 February 2024).
37. Rajapakshe, C.; Balasooriya, S.; Dayarathna, H.; Ranaweera, N.; Walgampaya, N.; Pemadasa, N. Using cnns rnns and machine learning algorithms for real-time crime prediction. In Proceedings of the 2019 International Conference on Advancements in Computing (ICAC), Malabe, Sri Lanka, 5–6 December 2019; pp. 310–316.
38. Alderliesten, K. Yolov3—Real-Time Object Detection. Available online: <https://medium.com/analytics-vidhya/yolov3-real-time-object-detection-54e69037b6d0> (accessed on 25 February 2024).
39. Chong, Y.S.; Tay, Y.H. Abnormal event detection in videos using spatiotemporal autoencoder. In *Advances in Neural Networks-ISNN 2017, Proceedings of the 14th International Symposium, ISNN 2017, Sapporo, Hakodate, and Muroran, Hokkaido, Japan, 21–26 June 2017*; Springer: Cham, Switzerland, 2017.
40. Atrey, J.; Regunathan, R.; Rajasekaran, R. Real-world application of face mask detection system using YOLOv6. *Int. J. Crit. Infrastruct.* **2023**. [CrossRef]
41. Sung, C.S.; Park, J.Y. Design of an intelligent video surveillance system for crime prevention: Applying deep learning technology. *Multimed. Tools Appl.* **2021**, *80*, 34297–34309. [CrossRef]
42. Jiang, B.; He, J.; Yang, S.; Fu, H.; Li, T.; Song, H.; He, D. Fusion of machine vision technology and AlexNet-CNNs deep learning network for the detection of postharvest apple pesticide residues. *Artif. Intell. Agric.* **2019**, *1*, 1–8. [CrossRef]
43. Forson, E. Understanding SSD Multibox—Real-Time Object Detection in Deep Learning. Available online: <https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab> (accessed on 25 February 2024).
44. Inception V3 Model Architecture. Available online: <https://iq.opengenus.org/inception-v3-model-architecture/> (accessed on 25 February 2024).
45. Sivakumar, P.; Jayabalaguru, V.; Ramsugumar, R.; Kalaisriram, S. Real Time Crime Detection Using Deep Learning Algorithm. In Proceedings of the 2021 International Conference on System, Computation, Automation and Networking (ICSCAN), Puducherry, India, 30–31 July 2021; pp. 1–5.
46. Phadtare, M.; Choudhari, V.; Pedram, R.; Vartak, S. Comparison between YOLO and SSD mobile net for object detection in a surveillance drone. *Int. J. Sci. Res. Eng. Manag* **2021**, *5*, 1–5.
47. Liu, K.; Zhu, M.; Fu, H.; Ma, H.; Chua, T.S. Enhancing anomaly detection in surveillance videos with transfer learning from action recognition. In Proceedings of the 28th ACM International Conference on Multimedia, Seattle, WA, USA, 12–16 October 2020; pp. 4664–4668.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.