

Efficient and Deployable Violence Detection using Transfer Learning Framework for Public Safety

Sanam Narejo^a, Shabana Hajano^a, Anoud Shaikh^b, Jibran Virk^a, Maheen Sheikh^a

^a Department of Computer Systems Engineering, Mehran University of Engineering and Technology, Pakistan

^b Department of Software Engineering, Mehran University of Engineering and Technology, Pakistan

* Corresponding author: Author Sanam Narejo, Email: Sanam.narejo@faculty.muet.edu.pk

Received: 06 February 2025, Revised: 11 August 2025, Accepted: 29 September 2025, Published: 01 October 2025

KEY WORDS

Smart city
Violence detection
Transfer learning
Closed-Circuit Television
Convolution Neural
Networks
Artificial Intelligence

ABSTRACT

One growing concern in urban areas is public safety in gathering spaces. Traditional Closed-Circuit Television (CCTV) monitoring systems depend almost entirely on human operators, making them prone to error due to fatigue and distraction. Due to the growing number of violent events, it is essential to develop automated intelligent systems to facilitate quick and accurate detection of potential violence, specifically in public spaces. This research aims to strengthen traditional surveillance systems by reducing human reliance in favour of AI-driven automation and implementing transfer learning (TL), a subset of Artificial Intelligence (AI), for developing an automated violence detection system. TL allows models trained on much larger, diverse datasets to be further fine-tuned for specific problems, e.g., violent actions in CCTV footage. Thus, the proposed system intends to efficiently classify violent and non-violent actions at a relatively lower computation cost. The final system is expected to be deployable in diverse public environments, thus contributing to faster incident response and improved public safety.

1. Introduction

Public safety has always been sensitive for the government, establishments, and citizens. Over the past decades, as the urban population density has increased, the need to make people feel at ease has also increased significantly. Aggressive behaviour in public spaces has become a significant issue, particularly in areas where large crowds gather. Closed-circuit television (CCTV) Surveillance systems have been installed widely to overcome this challenge. Video surveillance systems are continuously linked with a growing number of individuals and are publicly acknowledged by society. One of the significant benefits of using CCTV cameras is that they provide us with an essential layer of supervision, as these events can also be monitored afterwards and deter violent activities. Traditional CCTV systems often rely entirely on a human operator who monitors the video surveillance footage. Large-scale monitoring systems have a disadvantage in this state, as the human operators, while active and aware, are still prone to fatigue and can be distracted at times.

Apart from that, human supervisors are more likely to struggle to constantly maintain elevated levels of concentration, which in turn can lead to an increase in overlooking critical events. [1]. Violence is a type of behaviour that requires an urgent response, and detecting it through a traditional setup can lead to a delayed response because of manual monitoring. In the year 2020 alone, around 1.3 million violent crimes were recorded in the U.S. [2] In the United Kingdom, approximately 2.1 million cases were reported by March 2022. These numbers highlight traditional surveillance approaches' shortcomings and highlight the need for intelligent and efficient solutions.

New technological improvements have redefined the usage of monitoring systems with Artificial Intelligence (AI), mainly through transfer learning [3], which enables timely event detection and classification. These AI-based arrangements have reduced the previously complete reliance on human operators by delivering nonstop continuous assessment and reducing the liability of missing crucial events. Transfer learning has proven to be a

powerful tool for surveillance monitoring. It enables the models already trained on larger datasets to be fine-tuned to the specific needs of the tasks, in this case, detecting violent activities. Rather than starting from the beginning, transfer learning models use the knowledge it has acquired from the general dataset. By using these pre-defined models, the computational expense and the amount of data required for training and testing purposes are extensively reduced [4].

This study aims to eliminate the limitations of the existing surveillance monitoring system by automating the surveillance system that works by utilising the effectiveness of transfer learning models. Our work correlates with SDG 09 by promoting innovation and improvement in the public safety framework by using technologies such as Artificial Intelligence, namely Transfer Learning, a new concept given to surveillance systems.

2. Related work

Traditional video surveillance violence detection techniques are based on manual feature extraction. [5] [6,7] and machine learning classifiers [8,9,10,11]. These worked fine in austere environments but suffered from accuracy, adaptability, and computational efficiency problems in complex real-world scenarios. Nevertheless, they paved the way for more sophisticated systems. The study in [12] illustrates the necessity of identifying complex actions, including physical altercations, whereas traditional action recognition is mainly focused on simple daily-life behaviours, such as walking, jumping, etc.

Intermediate techniques in violence detection combine deep learning models with advanced architecture, which has enhanced the accuracy of violence detection and made it very efficient in moving above earlier models' spatial and temporal feature limitations, thereby further improving the performance and detection of complex cases [13]. By focusing on extracting features, the proposed technique in [14] has resolved the issue of computational expense and commonly occurring video processing challenges such as background noise, obstacles, and camera movements.

The researchers in [15] Presented a solution, which they called a "Spatiotemporal Encoder". The encoder extracts both spatial and temporal features from video frames. The framework utilises the VGG13 network as its spatial encoder, which is used for the extraction of feature maps from each individual video frame. The extracted maps are processed by the BiConvLSTM afterwards, which is used for temporal encoding in both directions (forwards and backwards). The proposed technique enables the model to get a better understanding of the contextual perspective of the

actions in video feeds. Following this, element-level maximisation is carried out to create a single representation of the entire video, which is later classified as violent or non-violent.

Previous methods utilised optical flow or conventional machine learning algorithms such as Support Vector Machines (SVM) [16]. The disadvantage of using these techniques was the reliability of handcrafted features for processing, and these were also likely to be overfitted due to the complexity of visual data. On the other hand, the combination of Conv2D and LSTM proposed in [17] presents a more reliable solution. The manual engineering of the features was not required as it automatically gathers and remembers the essential features from a large amount of data.

Recent advancements in violence detection use advanced deep-learning models. [19], real-time processing, and specialised architectures to enhance accuracy, speed, and robustness in detecting violent actions. Previous studies have shown that in many cases, only certain types of weapons were utilised as the focus points in automated detection systems, such as guns and knives, but in reality, various objects can be used as forms of weapons.

3. Deep Learning Models

Supervised learning, being a subfield of machine learning, trains the algorithms to discover relationships and patterns within labelled datasets. During the process of supervised learning, an algorithm is given a dataset that contains input-output pairs so that the algorithm can easily and accurately make predictions about new unseen data that it has been given. In a conventional supervised learning process, the dataset is divided into two: a training set and a testing set, which are used to train and test the model, respectively. During training, optimisation algorithms adjust the model's parameters to reduce the error between predictions and true labels, which is measured by a loss function.

In video-based tasks, it is important to analyse both spatial and temporal features to achieve reliable results. Videos, as we know, are not only a collection of still pictures but also a sequence of captured frames that contain both visual and dynamic context. Therefore, if the focus is only on spatial features, for instance, individual frames or movement patterns, the full picture cannot be depicted. Therefore, by combining the information gathered from both features, i.e., spatial and temporal, the complexity of real-life scenarios can be captured efficiently.

Spatial features are the static information that each individual frame of the video contains. These features include patterns like edges, textures, shapes, and color

distributions that help identify objects, people, and background elements in the scene. CNNs are typically employed to extract spatial features by scanning each video frame and learning the spatial structure of the objects and their interactions.

Temporal features mainly capture the dynamic information of a video by analyzing how these spatial features change over time across multiple frames. These features are important for understanding motion and activity patterns that occur over a series of consecutive frames, for example, in our case, punches or sudden movements that characterize violent actions. Temporal feature extraction typically involves specialised architectures like ConvLSTM2D, which can track the temporal evolution of spatial features, enabling the model to understand complex sequences of actions.

3.1 Transfer Learning: Convolutional Neural Networks (CNN)

In our violence detection research work, we utilised transfer learning, where pre-trained CNNs are used to extract spatial features from video frames. Rather than training a new model from scratch, we used common CNN architectures that were pre-trained on ImageNet, one of the largest annotated image datasets with millions of images and thousands of categories. These pre-trained models are optimised to detect many image features (edges, textures, and shapes) that can be helpful for our own training task. We tested the performance of three well-known CNN models, i.e., MobileNetV2, InceptionV3, and VGG16, to find out which one offers faster operation in our spatiotemporal violence detection architecture. Only the top-performing model is added to the final system. Each model and its integration into the architecture mentioned will be described in subsequent sections.

3.1.1 MobileNetV2

MobileNetV2 is a deep learning model that was specifically designed for mobile and edge devices with an emphasis on both speed and accuracy in the context of computer vision tasks. It improves its previous version, MobileNetV1, by adding inverted residuals and linear bottlenecks. By combining these, the model can zoom into essential information and reduce dimensionality by decreasing data volume.

One of the key techniques MobileNetV2 uses is a class known as depth-wise separable convolutions. Instead of performing a single convolution per layer across all channels in the input, this splits it into two easier steps—one for applying individual filters to each channel and another for combining them. This structured approach reduces the number of calculations, automatically increasing speed without compromising results.

MobileNetV2 is very efficient; it can be used on devices that are not powerful, such as real-time applications (like surveillance systems to detect some fights). Although this model is very small, the accuracy of its predictions is comparable to larger, heavily optimised models on standard benchmarks. It can also be fine-tuned with a few additional training iterations for tasks such as video content classification, including the identification of aggressive behaviour.

As can be seen in Fig.1, each standard convolution in MobileNetV2 is replaced by two simpler operations: 1. Depth-wise convolution applies individual filters to each channel rather than using a single filter per image with multiple input channels. 2. Pointwise convolution combines the outputs of these depth-wise convolutions through a 1x1 convolution. Additionally, MobileNetV2 introduces inverted residuals and linear bottlenecks, which are well-suited for capturing spatial relationships. The equation for depth-wise separable convolution is expressed in eq (1):

$$F_{out} = (k \times k \times C_{in} \times H \times W) + (1 \times 1 \times C_{in} \times C_{out} \times H \times W) \quad (1)$$

Where:

k is the kernel size

C_in is the number of input channels

C_out is the number of output channels

H and W are the height and width of the feature map.

This architecture was selected for its balance between accuracy and speed, making it suitable for our application.

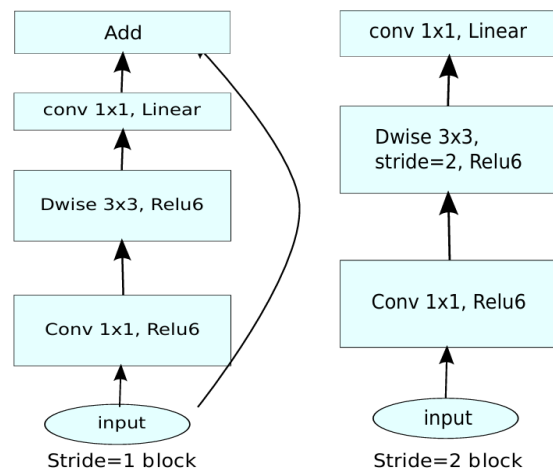


Fig. 1. Convolutional blocks for MobileNetV2 architecture

3.1.2 InceptionV3

InceptionV3 is a CNN that is a more accurate and better version than its ancestry Inception models with the same computational cost. It works by using a stack of inception modules that are multi-branch

architectures with convolutional filters working at multiple spatial scales within the same stage. This design allows the network to encapsulate various features in a variety of scales, which makes it better at detecting complex patterns from images. Furthermore, Inception V3 uses techniques such as factorised convolutions and dimensionality reduction to keep the parameter numbers in check, which will eventually help them be efficient and perform better.

InceptionV3 is an image classification (with a thousand classes) convolutional neural network architecture. It belongs to the family of Inception models, but it is optimised for feature extraction and works quite efficiently with a low computational cost. This architecture combines multiple sizes of convolutions within the same layer so that it can learn different features at various scales. This can be helpful in learning the rich representation of input images by the model.

The total computation cost of the Inception module is given by eq. (2).

$$F_{out} = (k_1 \times k_1 \times C_{in}) + (k_3 \times k_3 \times C_{in}) + (k_5 \times k_5 \times C_{in}) \quad (2)$$

Where k_1 , k_3 , and k_5 represent the different filter sizes used in the Inception module.

It can be seen in Fig. 2 that InceptionV3 architecture uses a combination of 1x1, 3x3, and 5x5 convolutions along with a pooling layer. The 1x1 convolutions perform two functions: they decrease the dimensionality of input feature maps (which reduces computational complexity) and also introduce non-linearities to our network. The output of these pathways is concatenated so that the network learns to extract features at different scales and spatial hierarchies. A major change in InceptionV3 is the introduction of auxiliary classifiers, which serve as additional paths for gradient propagation during training. These classifiers assist in better convergence and even empower the model to learn more explainable features.

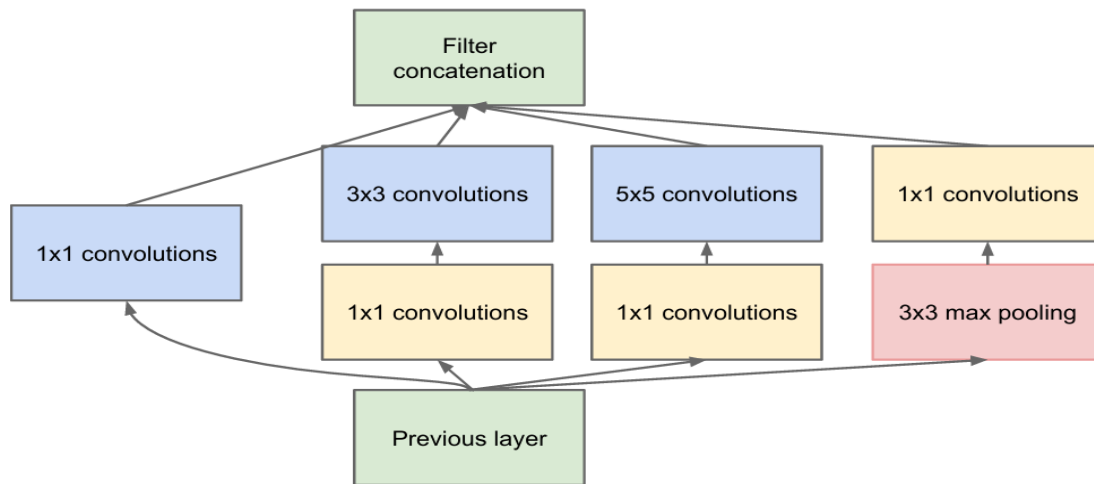


Fig. 2. Inception module with dimension reductions

3.1.3 VGG 16

VGG16 is a deep neural network known for its simple design and excellent performance in image recognition tasks. It has 16 layers with weights, including 13 convolutional and three fully connected layers. All convolutional layers in VGG16 use small 3x3 filters, which are stacked one after another to extract rich information from images. This approach makes it more explainable and has proven to be very good at preserving data details.

The subsequent step to these convolutional layers is an activation function, ReLU, in our case, which makes a neural network even more non-linear. Pooling layers following convolutional groups helps to reduce the size of the image computationally, reducing important features. Finally, there are fully connected layers at the end for a final classification. VGG16 is a

big model with approximately 138 million parameters, and it requires quite some resources to train, but its simplicity and reliability make it a really popular choice in many image recognition/image processing tasks.

Fig.3. represents the architecture of a CNN model, featuring multiple layers for feature extraction and classification. The network consists of several convolutional layers (L1 to L12) with ReLU activation functions and max-pooling operations, progressively reducing the spatial dimensions of the input while increasing the depth of feature maps. The final fully connected layers (L13 to L16) flatten the feature maps and perform classification, with a softmax layer for outputting probabilities over 1000 classes. This type of architecture is typically used for image classification tasks.

The primary equation for the convolution operation in VGG16 can be expressed as eq. (3):

$$F_{out} = f(C_{in} * W + b) \quad (3)$$

Where:

F_{out} is the output feature map

C_{in} is the input feature map

W represents the filter weights

b is the bias term

f is the activation function (ReLU in this case).

$*$ Denotes the convolution operation.

Although VGG16 is computationally more expensive than other models like MobileNetV2 and InceptionV3, it has a strong performance with high accuracy in various image classification tasks. This is because of its deep architecture, which is basically a deeper structure that allows it to learn complex patterns and spatial relationships between pixels in image data. VGG16 utilises a series of layers to perform successive stages (granularity in terms of detail) of feature learning, as each layer is able to detect details within an image with increasing levels of differentiation.

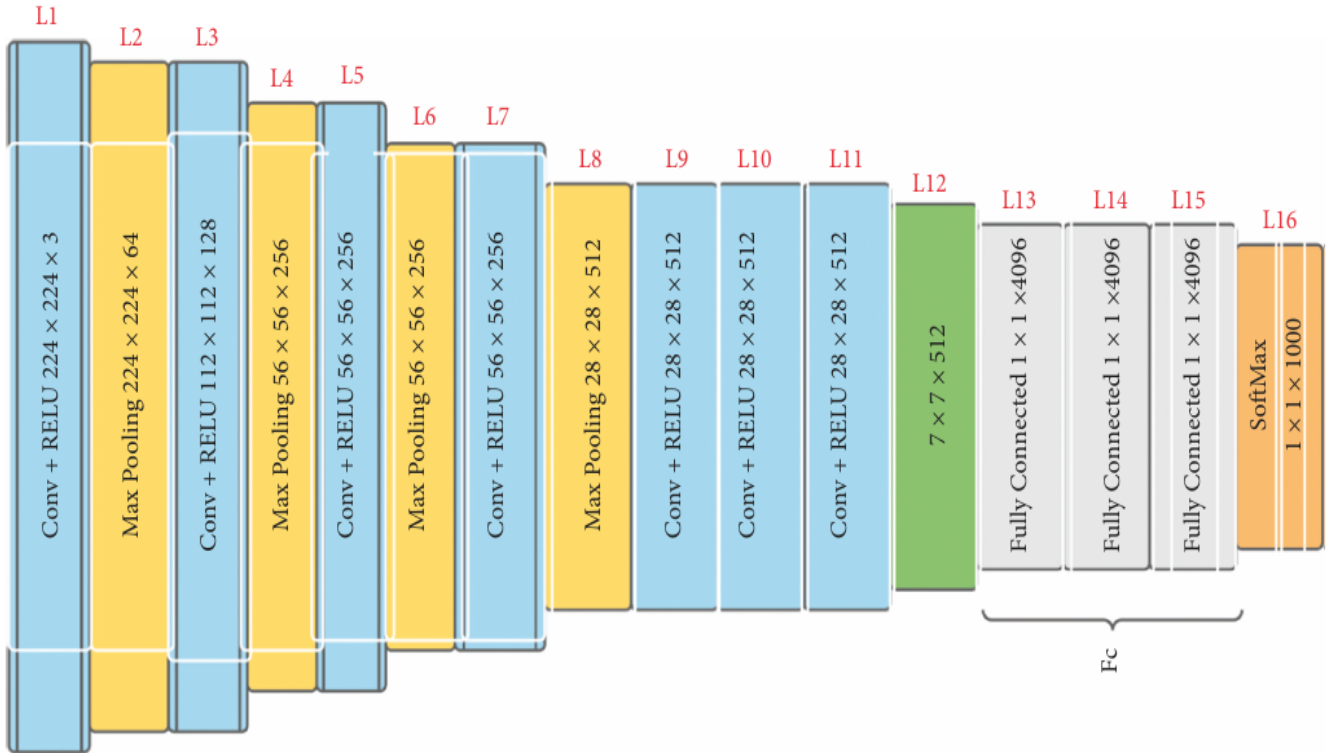


Fig. 3. The basic architecture of the VGG16 model

VGG16 architecture has a very high accuracy and can also store spatial features well, which produces satisfiable results and can be used for validating the effectiveness of the proposed model. We used VGG16 for the extraction of spatial features only and did not adopt it as a classifier for the reason that those extracted patterns can be classified correctly. VGG16 is employed in our evaluation to leverage its depth and detail-orientated structure, which is a good criterion for capturing high-level spatial features in designing the violence detection model.

In the context of video-based violence detection, spatiotemporal modelling is an essential functionality that facilitates the deep learning model with the spatial information present in each frame and temporal changes happening across multiple frames. This bi-

modal framework is necessary as spatial features are not enough to understand actions that evolve in time. Hence, to effectively combine the spatial and temporal dimensions within video data for a model to learn complex interactions between frames, we need another specialised strategy.

4. Methodology

4.1 Methodology

In machine learning, methodology means a clear-cut way of building, training, and testing a model where every stage is important for the final accuracy/reliability of prediction. This framework starts with dataset collection, which includes collecting a broad set of video sequences that are more diverse and reflect real-world dynamics. In order to

have better quality data and a more robust model, we pre-processed them by resizing, normalising, and augmenting the data. This was where the model oriented itself to a variety of conditions. By applying preprocessing and splitting the dataset into training, validation, and testing sets, we were able to ensure that our models were tested on an unbiased set of data, thereby reducing overfitting. The structured way of setting life to data things slowly laid a strong foundation for the model built for predicting violence correctly and consistently.

To develop our model, we used transfer learning by modifying pre-trained models (MobileNetV2, InceptionV3, and VGG16) that specialise in image spatial feature extraction. We emphasize that the base architectures of these models were retained without any structural modifications. Only the final classifier layers were customized for binary classification (Violence vs. Non-Violence). Moreover, we employed sophisticated layers like Bidirectional ConvLSTM2D to preserve the temporal dynamics in a manner that processes frames from both directions (i.e., forward

and backwards) that encapsulates all possible behaviour of an event. In our evaluation, the metrics such as accuracy, precision, recall, and F1-score helped us to evaluate its performance and confirm that it reached an accuracy level that is considered good enough for production. With this methodological framework, we proceeded to build a model that is able to detect violence using both physical spatial aspects of the environment coupled with similar temporal components.

The workflow diagram in Fig.4 describes the proposed Framework of violence detection using footage from CCTV. Data preprocessing includes frame extraction, resizing, and normalisation, along with splitting the data into the train and test sets. For example, VGG16, MobileNetV2, and InceptionV3 models have been used for training; metrics such as accuracy and precision are used for the model's evaluation. The best model is taken to predict the video content, which is then classified into normal or violent categories.

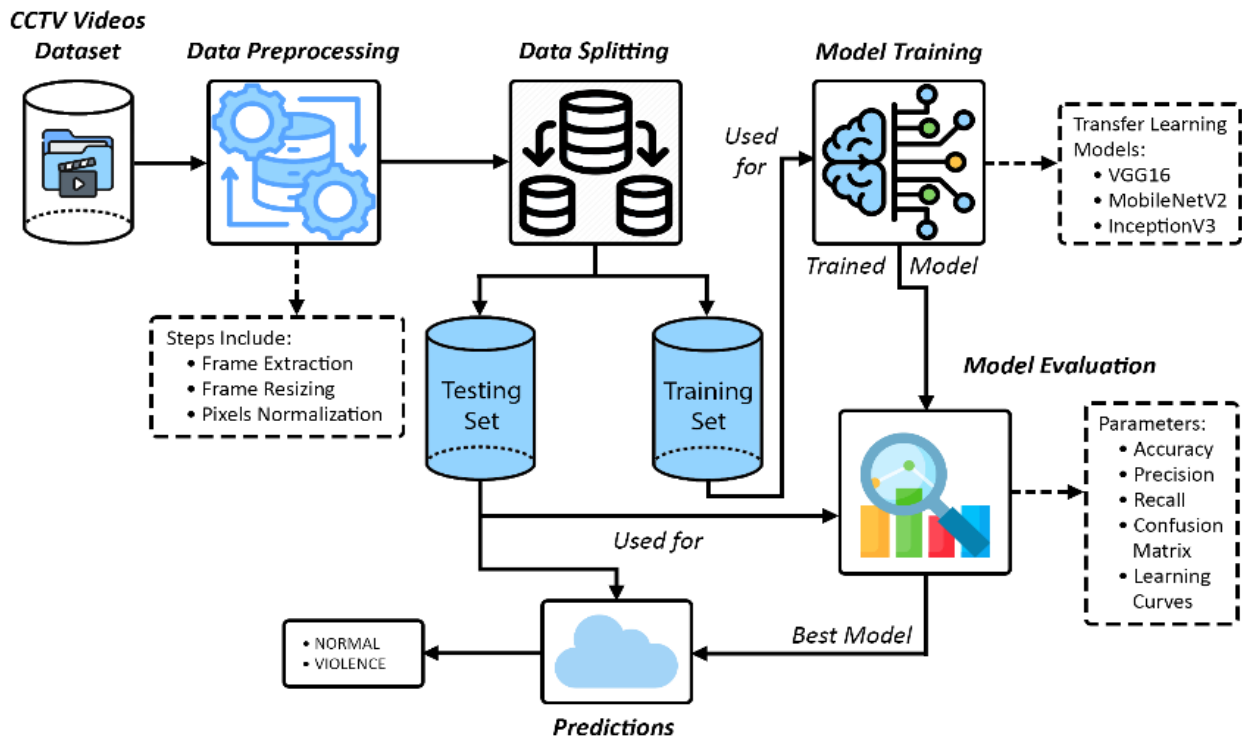


Fig. 4. Methodology Flow Diagram

4.2 Data Collection

We reviewed the available public data that better fit our project requirements. We analysed a couple of options and narrowed them down to two datasets, namely, the Smart-City CCTV Violence Detection Dataset (SCVD) [14] and the Real-Life Violence Situations Dataset. One of the benefits of considering SCVD as a labelled benchmark dataset source is that it contains closed-circuit live video footage, which aligns well with our model goal in surveillance

applications. In contrast, the RLVS dataset offers a rich training experience due to its variety of recorded scenarios in diverse conditions. When combined, these datasets span a significant number of environmental contexts and human activities, including many interactions ranging from mundane activity to physical altercations.

4.2.1 Smart-City CCTV Violence Detection Dataset (SCVD)

The Smart-City CCTV Violence Detection (SCVD) dataset [14], which is used in this project, has been tailored for all sorts of violence detection tasks in surveillance video. It includes three classes: Normal, Violence and Weaponized. In this study, however, the Weaponized class (weapon presence) was not included, and only two of these classes with practical values for detecting physical confrontations and fighting or normal activity were subsequently concerned namely Normal and Violence. Specific to the physical action recognition, we excluded the Weaponized class because it contains a different kind of complexity as compared to the above-mentioned Physical interaction and requires other training and evaluation criteria that were needed for.

The SCVD dataset is composed of 1088 Violent labelled videos and 1041 normal labelled videos, all of which are one second with a frame rate of 30 FPS. It has a defined train and set split, which is very useful for developing models as well as evaluating them. We sub-sampled 650 videos from each class to create a new dataset, which also has the same number of samples in both classes for training/validation. Both the unique labelling and frequency equalisation of this targeted dataset contribute towards ensuring that a model trained on it will reflect well on real-world physical violence detection surveillance scenarios both in terms of diversity and structure.

Table 1 below gives a description of the SCVD dataset with a total of 2129 videos, divided into two classes, Violence and Normal, with an equal number of videos, 650, in both classes. The length of each video is 1 second, with a frame rate of 30 FPS. This dataset is specifically prepared for surveillance-based violence detection with the specific purpose of identifying violent actions through surveillance cameras from video footage.

Table 1

Description of SCVD Dataset

Total Videos	2129
Classes Used	Violence, Normal
Videos per Class	650 per class
Video Length	1 second
Frames per Second (FPS)	30 FPS
Purpose	Surveillance-specific violence detection

4.2.2 The Real-Life Violence Situations Dataset (RLVS)

The Real-Life Violence Situations (RLVS) dataset was created to simulate violent behaviour observed during street fights and episodes of physical aggression in various settings. This dataset was presented in the paper "Violence Recognition from Videos Using Deep Learning Techniques". The dataset contains 1000 violent videos and 1000 non-violent videos collected randomly from YouTube and other public websites. Next, we are shown the different circumstances these concepts represent: The Violence class contains more raw fights in scenarios pertaining to the violence of everyday life, while the Non-Violence class features regular human activities such as walking, eating, or playing sports.

Because the videos in this data set have assorted lengths, we extracted 120 frames from each video as input to make sure that all samples had an equal number of frame inputs. We used these 1,000 videos per class and selected the first 650 videos from both classes for training and validation. This made sure that our model was trained on a variety of environments and situations to make it capable enough to identify any sign of violence in different kinds of scenarios. Another 350 videos that were left over were used for testing.

We balanced the number of samples for violence and normal (or non-violence) classes in each dataset to avoid class bias in training. This systematic dataset accumulation empowered the execution of a robust violent activity analysis algorithm adaptable to diverse real-life situational cases.

Table 2 shows that the RLVS dataset includes 2000 videos divided into Violence and Non-Violence classes, with 650 videos per class. The videos are 1 second long, but the FPS varies, with up to 120 frames being used in some cases. This dataset is intended for detecting violence under real-world, uncontrolled conditions.

Table 2

Description of RLVS Dataset

Total Videos	2000
Classes Used	Violence, Non-Violence
Videos per Class	650 per class
Video Length	1 second
Frames per Second (FPS)	Varying lengths (up to 120 frames used)
Purpose	Real-world uncontrolled video conditions

Table 3.

Dataset Characteristics Comparison

Datas et	No. of Vide os	FPS	Durati on Range	Classes	Annotati on Type
SCV D	2129	30	1 second	Normal , Violen ce	Manual
RLV S	2000	Varyi ng (up to 30 FPS)	1 second	Non- Violen ce, Violen ce	Manual

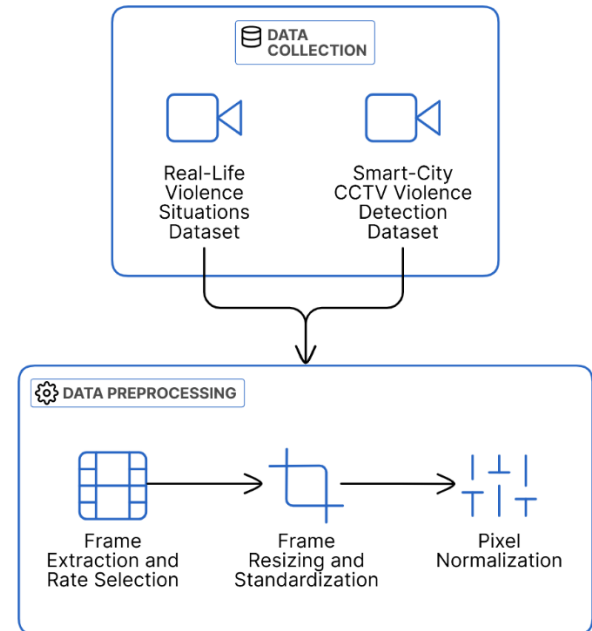
The SCVD dataset includes surveillance-type videos of uniform 30 FPS and 1-second length and duration, respectively, and thus is best suited for controlled violence detection applications. In contrast, the RLVS dataset is aggregated from various real-world sources and shows uneven frame rates and real-world background noise. These variations call for strong preprocessing and normalization to facilitate even feature extraction across both datasets.

4.3 Data Preprocessing

Data preprocessing is an important step in the machine learning pipeline. Preprocessing is exactly what it sounds like: the transformation of raw data to an understandable format so that our models can more easily pull out insights, trends, and other relevant information. As raw data is frequently messy and incomplete or inconsistent, preprocessing improves the quality of available training examples by cleaning them up to enable a better representation for learning suitable patterns in this phase. Preprocessing simply consists of correcting imprecise data and formatting the records into a unified format.

For video data or image-based data, preprocessing is very domain-specific. The images are often resized, cropped, or converted into grayscale in order to make the computational load lighter and improve model efficiency. In video analysis, the individual frames may be recovered and corrected for consistency over samples. Therefore, the preprocessing translates into preparing both cleaning and structuring data but also in increasing diversity, or even more so that samples end up being representative of what models will learn, making such a step absolutely necessary if you are to construct precise machine learning systems.

Fig.5. depicts the data collection and preprocessing pipeline for a violence detection system. It uses two datasets: the Real-Life Violence Situations Dataset and the Smart-City CCTV Violence Detection Dataset. The preprocessing involves extracting frames at a specific rate, resizing them to a standard dimension (e.g., 224x224 pixels), and normalising pixel values to prepare the data for efficient model training.

**Fig. 5.** Data Preprocessing pipeline

4.3.1 Frame Extraction and Rate Selection

Frame extraction plays an essential role, where spatial and temporal information are both very crucial features. Our approach, which divides videos into single frames, allows the model to extract high spatiotemporal context from the visual content of each individual frame. In addition to this, it also captures crucial motion dynamics by examining transitions between each frame. This stage transforms the videos into a state that is appropriate with convolutional neural networks for detecting key patterns indicative of violence. Additionally, it allows the application of preprocessing techniques at each frame level (resizing and normalisation) to make the data more compatible with being fed into the network.

As far as the frame rate goes, we used the videos at 30 fps for frame extraction. This rate balances capturing enough details with computation efficiency. If the frame rate is too low, it could cause crucial movements to be missed, especially during hyper-violence-filled scenes with rapid and jerky actions. At the same time, a higher frame rate would just add additional loads on memory/processing without bringing substantial benefits. Operating at 30 Frames per Second (FPS) allows the model to identify fast actions and drastic changes from a reasonable amount

of data, thus allowing it to capture enough temporal patterns to detect violence correctly.

The bar chart in Fig.6. shows the distribution of videos based on their frame rates, rounded to the nearest 5 FPS. Most videos in the dataset have a frame rate of 30 FPS, with over 3000 instances. Frame rates around 25 FPS are the second most common, while other frame rates like 10, 20, and 35 FPS appear infrequently. This distribution highlights the dominance of 30 FPS videos in the dataset.

One often overlooked aspect is the importance of retaining a consistent framerate between different datasets to make sure that those extracted temporal features are comparable and meaningful for your model. In the presence of mixed frame rates in datasets, temporal information might be inaccurate and could lead to incorrect estimations. As a result, following the same frame rate standard with 30 FPS for every video in the SCVD and RLVS datasets allows us to have an invariant temporal resolution so that it makes learning easier on both systems while improving spatiotemporal analysis robustness from more diverse sources. This uniform separation of the representations implies that any frame sequence will provide a similar temporal experience and, hence, better model performance and generalisation.

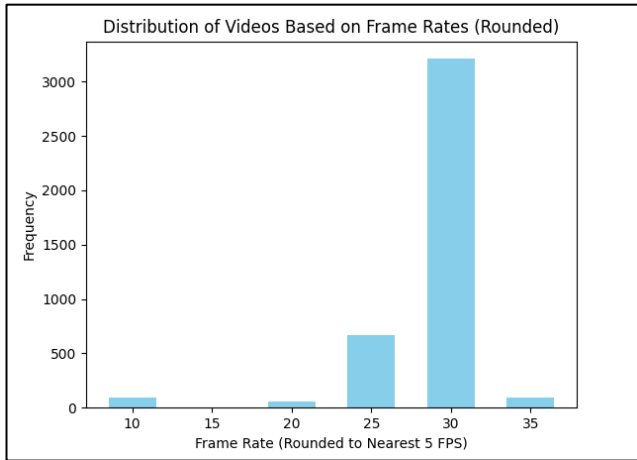


Fig. 6. Frame Rate Distribution

4.3.2 Frame Resizing and Standardization

The extracted frames from the dataset are resized to 224x224 pixels with three channels, as these are the standard input dimensions used by all CNN architectures we use. Sizing frames to a standard resolution is necessary for consistency and compatibility across the model layers, as they expect inputs of uniform shape and size. This required preprocessing step allows the model to work with visual data consistency across all frames.

Figure 7 illustrates the distribution of video resolutions in the dataset. The majority of videos have a resolution of 1280x720, followed by 224x224, which was likely resized for model compatibility.

Resolutions such as 640x360 and 854x480 are less frequent, with a small number categorised as “other”. This indicates a predominance of high-definition (HD) videos, with some preprocessing applied for uniformity.

However, resizing the original frame size greatly improves computational efficiency, not only by standardising it. Reducing the size of a pixel means less memory and computational load, which allows you to train and test your model in hardware-limited environments.

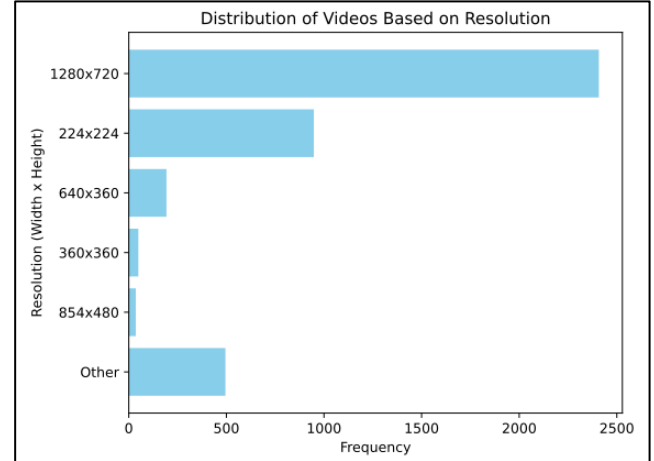


Fig. 7. Resolution Distribution

4.3.3 Pixel Normalization

In this study, we obtain a normalisation of pixel values by dividing each single pixel with 255 where the original range for pixels was from [0 to 255] and it now ranges between [0, 1]. This is a necessary step in the pre-processing to make it suitable so model learning can take place on images that are normalised between 0 and 1. We also normalise the values of pixels by mapping every value to [0, 1]. The reason is that it allows us to deal with numerical instability in a straightforward way and helps our model process and interpret image data much better. This also results in more convex gradients during training, leading to better convergence of the model by reducing variance when minimum and maximum values for data are set at each scale.

Transfer learning models like MobileNetV2, InceptionV3, and VGG16 require normalisation as they were initially trained on normalised image data. To be able to transfer the learned feature representations effectively, it is important that we achieve a similar input distribution for our dataset. If we deploy the CNN, which is pre-trained on ImageNet, into one of the datasets used, with input values aligning well, then these pre-train weights can generalise much better and make both performance and stability increase. Not only is it a standard pre-processing step, but it is also important to take advantage of the full capabilities of pre-trained models in transfer learning.

4.3.4 Efficient Frame Management with Data Generators

Memory management with the help of data generators reduces memory overheads while training your model. While loading the entire dataset of video frames into memory will fit in some use cases, it can quickly lead to a very large amount of memory consumption. This is where data generators come in handy by dynamically loading video frames as needed. This is quite useful when you need to work with large datasets because trying to store all the frames at once would not work as it will use more memory resources and give errors.

They enable the loading of data instantly by specifying that only the required frames for each batch during training be loaded. The generator loads the corresponding frames from the disk, preprocesses them, and then feeds them into the training pipeline as the model processes each batch. This way, we can avoid the scalability issue where the memory footprint grows so that larger and more diverse datasets can be utilized with fast system performance.

This also improves training efficiency, as the model can keep sampling new data without having to wait until everything loads up front. The model will concentrate just on learning from the incoming frames, thereby making efficient use of resources by keeping the data pipeline lean and smooth. Data generators are a common strategy that helps both avoid memory issues and provide effective training for complex models on large-scale video datasets.

4.4 Data Splitting

Machine learning requires the dataset to be split into training, testing, and validation sets to make sure that the model is robust enough to generalize. The training set is what the model learns from; the validation set is for us to evaluate and fine-tune performance while we train our model on; testing data is used to provide an unbiased evaluation of a final trained method. In this project, a systematic way of splitting the data was used to allow any model as much opportunity for learning.

The main dataset is used to train a model, and it contains 75% of the predefined videos, which means that in both datasets, for instance, the Normal set and Violence set are about 487 videos from each class, giving us 974 per violence or normal. The most important task was to balance our classes between Normal and Violence, allowing us to fix a class imbalance so that it does not favor the model's learning with performance degradation on minority classes.

A subpart of the data is validation data. This is used to check for generalization of modeling during

training when updating hyperparameters and architecture. We used 25% of the videos as a validation set from both datasets, which turned out to be $(163 \text{ samples per class}) \times 2 = 326 \text{ samples per class}$ in each dataset. These samples were picked from those selected 650 sample videos belonging to each original class. The addition of a regular evaluation on the validation set prevented overfitting by showing where the performances in terms of how much loss got saturated, therefore tuning up to better adapt to new unseen data.

It is a different part of the dataset that was unseen for any model during training and validation. The testing set was 350 videos per class from RLVS, and for SCVD, the test split had an imbalanced division of normal (164 videos) to violent (378 videos).

Table 3 outlines the allocation of videos into training, validation, and testing sets for both datasets. The Training Set consists of equal numbers of normal and violent videos from each dataset, with a combined total for both datasets. The Validation Set follows a similar distribution, with a balanced number of normal and violent videos. The Testing Set includes videos from both categories, ensuring comprehensive evaluation. This distribution maintains balance across all sets to ensure effective model training, validation, and testing.

Table 3

Distribution of Dataset Splits

Dataset Split	Dataset	Normal Videos	Violence Videos	Total Videos
Training Set	SCVD	487	487	974
	RLVS	487	487	974
	Total	974	974	1948
Validation Set	SCVD	163	163	326
	RLVS	163	163	326
	Total	326	326	652
Testing Set	SCVD	164	378	542
	RLVS	350	350	700
	Total	514	728	1242

4.5 Model Architecture

The architecture of our proposed model is to concatenate pre-trained CNN with sequential layers for spatial and temporal patterning, respectively. We used three pre-trained CNN backbone networks, MobileNetV2, InceptionV3, and VGG16 for feature extraction as they are able to capture strong spatial features by virtue of being trained on large-scale datasets initially. The use of pre-trained weights from these models for transfer learning was confirmed to be effective, hence improving efficiency and maintaining essential frame details.

In order to effectively model temporal changes over time, we designed the input to be 15 consecutive frames from each video. By using such a sequence length, the model can capture fine-grained motion cues and event progressions over short time spans that are essential to correctly detecting violent activities.

In feature extraction, we added ConvLSTM2D layers to capture the time aspect across frames. Such layers capture moving cues, which is critical in the recognition of violent activities. Bidirectional LSTM layers have been added to the second stack of layers in order to provide more complicated temporal understanding and control on both sides, thus ensuring it covers all movement information inside the video.

Time-distributed layers were applied for frame-wise processing, allowing the CNN to consume each frame in sequence before temporal features are computed. This setup enabled us to ensure spatial consistency among the details extracted per frame, which in turn boosted the capacities of our temporal layers.

The input has 15 video frames, each sized 224x224x3. In the Feature Extraction step, the transfer learning model (MobileNetV2, InceptionV3, VGG16) is used with Time Distributed to handle each frame one by one, then MaxPooling2D and Batch Normalization are applied to make the size smaller and help the training stay stable.

The Temporal Analysis part uses MaxPooling3D to downsize the time dimension. It also employs a Bidirectional ConvLSTM2D to understand those frames' two-way time relationships.

In the Classification stage, all the features are flattened. They pass through a dense layer with ReLU activation and then have a Dropout layer to avoid overfitting. The output is one unit activated through Sigmoid that gives the possibility of violent activity existing within the video sequence. In this way, there has been an effective integration of both spatial and temporal processing while detecting violence.

The first stage, model training, is the essence of any machine learning pipeline, as it teaches the network how to identify patterns and generate predictions using only its underlying architecture on top of a set of data known as training samples. In this stage, the model calculates and adjusts its parameters iteratively based on the loss function, which is calculated from the difference between true labels and predicted results. Proper training is necessary so that the model can ideally be generalised to new, unseen data in order to have reliable performance on deployed systems.

The model was trained in this study with an earning sixate of 1 time 10 to the minus six end superscript, which is an important hyperparameter for determining

the step size during weight updates. A lower learning rate can slow down the process of converging to an optimal solution more gently instead of overshooting minima due to specifically sharp valleys in the loss landscape. We used a batch size of 2 videos, one from the SCVD dataset, which has 30 frames, while the other is from the RLVS dataset, i.e., trusting it to be at least four times longer than the previous (120 frames). This was a resource-friendly setting in that 150 frames were processed per iteration, which is how many the model required for 15 consecutive frame inputs.

The training took place in 20 epochs, and at each epoch, we checked the performance of our model by validation loss. An early stopping pre-processing method was used for the purpose of generalization improvement and to prevent overfitting. This meant that training will be stopped for the model when there is no improvement in validation loss over five successive epochs, thereby storing weights of the best

-performing part of the earlier phases. These strategies are important to improve model performance without degrading its ability to learn new data points.

Furthermore, model checkpointing was used to secure any training issues. That way, the model would be saved at the end of every epoch on that iteration if it had hit minimum validation loss. This approach ensured a strong fallback by saving the model state with the best top performance, thereby not letting suboptimal weights affect its prediction.

Table 4

Summary of Hyperparameters

Hyperpar ameter	Value	Description
Learning Rate	1×10^{-6}	Controls the step size during weight updates.
Batch Size	2 videos (150 frames)	Number of training samples used in one iteration.
Epochs	20	The total number of passes through the training dataset.
Early Stopping	Patience: 5 epochs	Stops training when validation loss does not improve for five epochs.
Checkpo int Saving	Monitors: val_loss	Saves the model at the epoch with the best validation loss.

Table 4 outlines the training process that was guided by a set of key hyperparameters to ensure effective learning and prevent overfitting. A small learning rate facilitated gradual updates to the model, while the batch size determined the number of samples processed in each iteration. The training was set to run for a defined number of epochs, with early stopping

applied to halt training if performance on validation data ceased to improve. Additionally, checkpoint saving was used to retain the best-performing model based on validation loss.

4.6 Video Processing Workflow

The process begins with Frame Extraction, where the uploaded video is converted into individual frames. It is important as it enables the model to make predictions on every frame, providing fine temporal information which hints at violent actions.

Following the frame extraction part comes the predicting and labeling of frames. This stage gives the model-generated labels to the frames. The frames that have been identified as "Normal" get green print, while those containing "Violent" actions get red print. All this colour-coded labelling creates bright visual feedback on each frame about what part of the video is normal and which is violent.

At this point, the labelled frames are again merged. In this case, the labeled frames will be merged sequentially in the form of a video; however, it will produce a new video that displays a continuous format of labeled predictions. This final video captures the detection results very effectively.

The output video file is then converted using FFmpeg into the H.264 format compatible with web-based video players. That means the processed video, once created, can be easily viewed and accessed by the end-user directly from the web application.

4.7 Experimental Setup

The model was implemented using Python 3.10, TensorFlow 2.10, and Keras 2.10. Experiments were conducted on both Google Colab and Kaggle platforms. Colab provided 12.7 GB CPU RAM and 15 GB GPU RAM; Kaggle offered 29 GB CPU and 16 GB GPU RAM.

5. Results and Discussion

The aim is to develop a model with transfer learning and spatiotemporal modelling that efficiently separates violent from non-violent events in diverse environments. In this section, we present the performance evaluation of our developed ML-based violence detection models and discuss their performance and limitations. The quality of each model is tested on the basis of how capable these models are for detecting violent incidents with key matrices. The metrics that are employed are accuracy, precision, recall, and the F1 score. Some visualisations, such as a learning curve and a confusion matrix, show us how much the models have learned. Furthermore, we evaluate the three Transfer Learning architectures (MobileNetV2, InceptionV3, and VGG16) to identify which model will work best

with our setup. It ends with the assessment of misclassifications and the identification of challenges encountered during training.

5.1 Discussion on Negative Transfer

Negative transfer is possible where inconsistent features in the source and target domains lead the learning process astray. In our situation, SCVD and RLVS datasets differ in context, illumination, and camera views. To minimize feature inconsistency, we utilized preprocessing methods like resizing, frame normalization, and class balancing. These measures ensured the CNN backbone learned domain-neutral features and therefore reduced the likelihood of negative transfer and enhanced generalization.

5.2 Experimental Results

The learning rate (η) is one of the most impactful hyperparameters, which directly drives the model to its training process. Thus, it controls the number of adjustments to its weight during the optimisation procedure. A high learning rate accelerates convergence but may overshoot the optimal solution or cause instability. A low learning rate implies stable updates but can cause slow training and suboptimal results. To ensure optimal model performance with stability across the training process, a suitable balance needs to be established. A set of experiments was conducted with four different values of learning rate: 1×10^{-7} , 5×10^{-7} , 1×10^{-6} , and 5×10^{-6} . All the models, including MobileNetV2, VGG16, and InceptionV3, were trained with uniform conditions, and their respective accuracies are noted for each learning rate. The results of the experiments are summarised in Table 5.

Table 5

Model Accuracies at Different Learning Rates

Learning Rate	MobileNetV2	InceptionV3	VGG16
1×10^{-7}	79%	77%	76%
5×10^{-7}	82%	79%	78%
1×10^{-6}	86%	84%	81%
5×10^{-6}	83%	80%	79%

As observed in Table 5, the accuracy depends upon different learning rates. The smaller learning rates of 1×10^{-7} and 5×10^{-7} led to moderate accuracies for all three models. The highest accuracies were gained with the learning rate 1×10^{-6} , where MobileNetV2 reached 86%, VGG16 reached 81%, and InceptionV3 reached 84%. On the other hand, 5×10^{-6} proved to be a larger learning rate that found a slight decrease in the performance of all the models. The experiments carried out, therefore, give a comparative view of how different learning rates affect model performance, thus

showing the relation between learning rate and the exactness of models.

An accuracy score is one of the key metrics here that will allow us to know how well our model identifies violent events from non-violent events. It calculates the proportion of positive true cases over all tested instances. High accuracy implies the model can generalise well over a diverse set of situations within our dataset, i.e., various lighting and miscellaneous background activity. Accuracy is not the sole means of the performance of the model, especially in case prediction has imbalanced classes. To make sure we have the complete scope, other metrics such as Precision and Recall (or the F1-score) are used in order to better assume how well the technique adopts forceful events, indicating its real capacity. All these metrics are elaborated on in the next sections, trying to provide an explanation for a more comprehensive evaluation of how effective our model is.

Table 6

Accuracy Measures

Metrics	MobileNetV2	InceptionV3	VGG16
Accuracy	86%	84%	81%
Precision	85%	84%	80%
Recall	85%	83%	81%

Table 6 presents a performance comparison of three models, MobileNetV2, InceptionV3, and VGG16, which were evaluated using accuracy, precision, and recall metrics. Among the models, MobileNetV2 demonstrates the highest performance, achieving an accuracy of 86%, a precision of 85%, and a recall of 85%. Therefore, MobileNetV2 is considered the "best model" in Figure 4, which shows the workflow where the highest-performing model is used for final classification. InceptionV3 comes next, with slightly lower scores accuracy at 84%, precision at 84%, and recall at 83%. Finally, VGG16 shows the lowest performance, with an accuracy of 81%, a precision of 80%, and a recall of 81%. Overall, this indicates that MobileNetV2 outperforms the other models in this comparison.

A confusion matrix is a very powerful way of illustrating the performance of a classification model by showing actual and predicted value distribution. The confusion matrix gives us four important performance metrics of a binary classifier task such as violence detection, namely, True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). All the metrics have their significance in measuring how accurate and reliable our model is.

The confusion matrix is in Fig.8. shows that MobileNetV2 correctly predicted 420 cases of

"Normal" and 646 cases of "Violence", while it misclassified 94 "Normal" cases as "Violence" and 82 "Violence" instances as "Normal". The matrix shows a strong overall classification performance with relatively low misclassification rates for both categories.

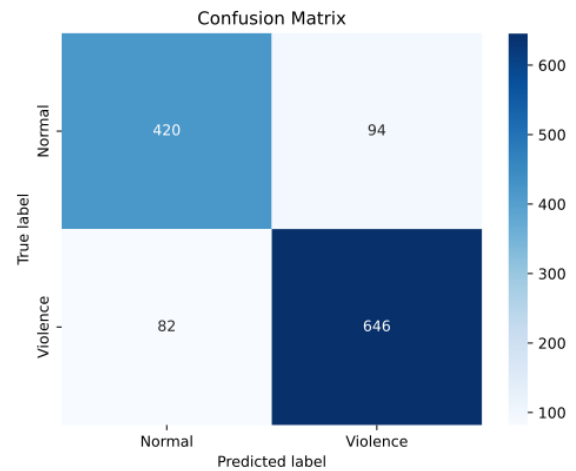


Fig. 8. Confusion Matrix for MobileNetV2

Fig.9. shows that InceptionV3 correctly predicted 405 instances of "Normal" and 638 instances of "Violence," while it misclassified 109 "Normal" instances as "Violence" and 90 "Violence" instances as "Normal". It indicates that there is a slightly higher misclassification rate for the "Normal" category compared to the "Violence" category.

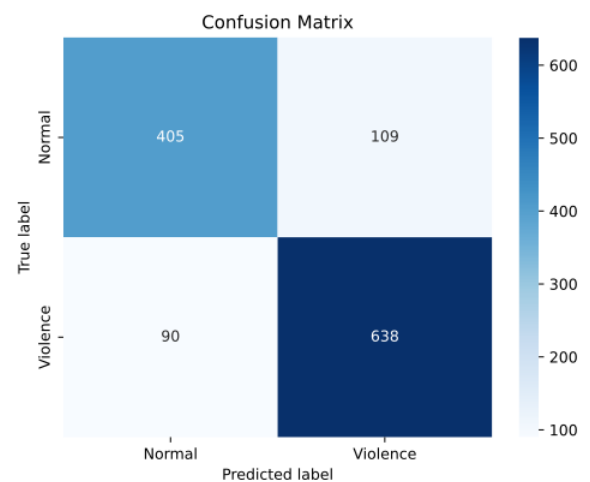


Fig. 9. Confusion Matrix for InceptionV3

Fig.10. represents the performance of the VGG16 model in classifying the categories Normal and Violence. The model correctly predicted 410 instances of "Normal" and 597 instances of "Violence," but it misclassified 104 "Normal" instances as "Violence" and 131 "Violence" instances as "Normal." While the model demonstrates reasonable performance, there is a noticeable imbalance in the misclassification, with more errors in predicting the "Violence" category compared to "Normal."

Our focus was to develop an automated model that can distinguish violent vs non-violent activities using transfer learning models on surveillance video cameras. Despite the fact that it reliably identified violent actions, a few issues popped up while developing this system, as much of the process relied on computations and models used in both training and testing amidst different environments. The Smart-City CCTV Violence Detection (SCVD) dataset with realistic CCTV footage was both large and challenging to use. It was difficult to load it into memory because the available cloud platforms had limited memory. Because of this, we had to read the data in smaller and slower batches. Further, since some videos were just everyday moments being recorded by bystanders and others shared accounts of violence in progress or afterwards, this mix could have caused additional issues. While Kaggle and Google Colab allow for the use of GPUs, using free versions had restrictions either on memory or how long a session could run. Kaggle has 29GB of CPU RAM and 16 GB of GPU RAM, while on Colab, you have only 12.7GB of the former and 15GB of the latter. That wasn't enough to process a large dataset directly at once. To overcome this, a data generator was employed, which helped but made the training process slower overall. Training on a large video dataset is extremely time-consuming, and the memory boundaries made it even worse. The session limit of 12 hours at Kaggle was not enough to finish one complete training process. We solved this by using model checkpointing that saved the state of the model after each training cycle so it could continue from there. However, this still caused delays, as training had to be spread over several sessions. Models were initially deployed on Google Cloud using their free trial. Such only allowed for access to the CPU without GPU acceleration; this really reduced the models' inference time. Instead, models trained on GPUs do inference much faster, but without access to a GPU, the project was running inference times up to 25x slower than what was expected. This really impeded the production of real-time violence detection because the models could not have a video frame processed in time to act.

5.3 Ablation Study

In order to comprehend the contribution of individual parts of our architecture, we performed an ablation study in two stages:

1. **CNN Backbone Comparison:** We substituted MobileNetV2 with InceptionV3 and VGG16 individually. Results (presented in Table 6) show that MobileNetV2 has the highest accuracy (86%) because of its lightweight nature and efficient feature extraction on edge-like datasets.

2. **Temporal Layer Comparison:** We replaced the ConvLSTM layers with regular LSTM. This led to a decrease of ~5% in accuracy, verifying the significance of spatiotemporal context modeling through Bidirectional ConvLSTM.

These results reaffirm that both spatial backbone choice and temporal modeling architecture play a significant role in influencing overall system performance.

6. Conclusion

This work focused on developing an automated violence detection system to analyse surveillance footage to identify physical altercations. The study addressed a critical problem in traditional CCTV setups: the reliance on human operators for monitoring, which often leads to lapses in detecting crucial events. By leveraging transfer learning, the study compared three pre-trained models—MobileNetV2, InceptionV3, and VGG16—to detect violence with high accuracy while optimising computational efficiency.

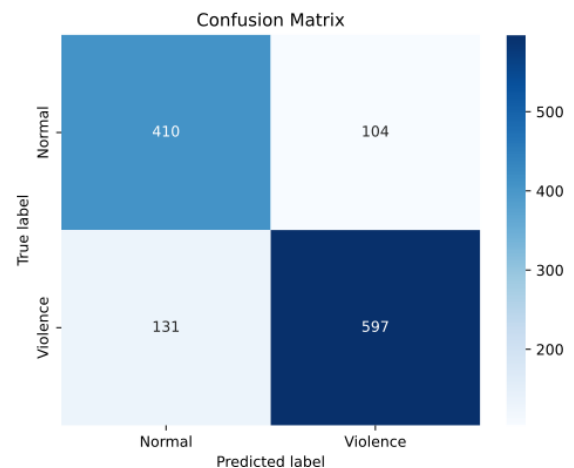


Fig. 10. Confusion Matrix for VGG16

In particular, both the SCVD dataset and the RLVS dataset were used to train and test the system. This setup offers a base of videos regarding the CCTV environment, involving both violent and non-violent scenarios, and the other contains some real-life violence situations filmed in all types of environmental conditions. It was implemented with TensorFlow and Keras, mainly using Python. Hardware restrictions are a concern, so a data generator was made to load the data one small batch at a time as it trains in a memory-friendly fashion.

While this study demonstrates the potential of transfer learning for violence detection from CCTV footage, there are several areas for improvement and future research that could further enhance the system's effectiveness and real-world applicability.

MobileNetV2 was selected because of its efficiency. Even more optimisation of models for the low-resource environment (like reducing model size, pruning networks, or implementing quantization) would make the system more deployable in real-world surveillance systems with limited computational power.

6. References

- [1] Y. Bi, D. Li, and Y. Luo, "Combining Keyframes and Image Classification for Violent Behavior Recognition," *Applied Sciences*, vol. 12, no. 16, p. 8014, Aug. 2022, doi: 10.3390/app12168014.
- [2] "US Homicides Increased Nearly 30% in 2020, FBI Reports." Accessed: Feb. 02, 2025. [Online]. Available: <https://www.voanews.com/a/us-homicides-increased-nearly-30-in-2020-fbi-reports-/6247529.html>
- [3] R. A. Taha, A. A. H. Youssif, and M. M. Fouad, "Transfer learning model for anomalous event recognition in big video data," *Scientific Reports 2024 14:1*, vol. 14, no. 1, pp. 1–14, Nov. 2024, doi: 10.1038/s41598-024-78414-2.
- [4] A. S. Keçeli and A. Kaya, "Violent activity detection with transfer learning method," *Electron Lett*, vol. 53, no. 15, pp. 1047–1048, Jul. 2017, doi: 10.1049/el.2017.0970.
- [5] A. J. Naik and M. T. Gopalakrishna, "Violence Detection in Surveillance Video-A survey," *International Journal of Latest Research in Engineering and Technology (IJLRET) www.ijlret.com || PP*, pp. 11–17, [Online]. Available: www.ijlret.com
- [6] K. Lloyd, P. L. Rosin, D. Marshall, and S. C. Moore, "Detecting violent and abnormal crowd activity using temporal analysis of grey level co-occurrence matrix (GLCM)-based texture measures," *Mach Vis Appl*, vol. 28, no. 3–4, pp. 361–371, May 2017, doi: 10.1007/s00138-017-0830-x.
- [7] Y. Watanabe, M. Okabe, Y. Harada, and N. Kashima, "Real-World Video Anomaly Detection by Extracting Salient Features in Videos," *IEEE Access*, vol. 10, pp. 125052–125060, 2022, doi: 10.1109/ACCESS.2022.3224952.
- [8] F. D. M. De Souza, G. C. Chávez, E. A. Do Valle, and A. D. A. Araújo, "Violence detection in video using spatio-temporal features," *Proceedings - 23rd SIBGRAPI Conference on Graphics, Patterns and Images, SIBGRAPI 2010*, pp. 224–230, 2010, doi: 10.1109/SIBGRAPI.2010.38.
- [9] A. A. Sodemann, M. P. Ross, and B. J. Borghetti, "A review of anomaly detection in automated surveillance," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 42, no. 6, pp. 1257–1272, 2012, doi: 10.1109/TSMCC.2012.2215319.
- [10] X. Fan, L. Xu, X. Zhang, and L. Chen, "The research and application of human detection based on support vector machine using in intelligent video surveillance system," *Proceedings - 4th International Conference on Natural Computation, ICNC 2008*, vol. 2, pp. 139–143, 2008, doi: 10.1109/ICNC.2008.315.
- [11] J. Albusac, J. J. Castro-Schez, L. M. Lopez-Lopez, D. Vallejo, and L. Jimenez-Linares, "A supervised learning approach to automate the acquisition of knowledge in surveillance systems," *Signal Processing*, vol. 89, no. 12, pp. 2400–2414, Dec. 2009, doi: 10.1016/J.SIGPRO.2009.04.008.
- [12] F. Kulsoom, S. Narejo, Z. Mehmood, H. N. Chaudhry, A. Butt, and A. K. Bashir, "A review of machine learning-based human activity recognition for diverse applications," *Neural Comput Appl*, vol. 34, no. 21, pp. 18289–18324, Nov. 2022, doi: 10.1007/S00521-022-07665-9.
- [13] E. Bermejo, O. Deniz, G. Bueno, and R. Sukthankar, "Violence Detection in Video Using Computer Vision Techniques." [Online]. Available: <http://visilab.etsii.uclm.es/>
- [14] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, Sep. 2014, Accessed: Feb. 03, 2025. [Online]. Available: <https://arxiv.org/abs/1409.1556v6>
- [15] P. Zhou, Q. Ding, H. Luo, and X. Hou, "Violence detection in surveillance video using low-level features," *PLoS One*, vol. 13, no. 10, p. e0203668, Oct. 2018, doi: 10.1371/JOURNAL.PONE.0203668.
- [16] J. Vidhya and R. Uthra, "Violence detection in videos using Conv2D VGG-19 architecture and LSTM network," 2021.

- [17] S. Narejo, B. Pandey, D. Esenarro Vargas, C. Rodriguez, and M. R. Anjum, "Weapon Detection Using YOLO V3 for Smart Surveillance System," *Math Probl Eng*, vol. 2021, no. 1, p. 9975700, Jan. 2021, doi: 10.1155/2021/9975700.
- [18] T. Aremu, L. Zhiyuan, R. Alameeri, M. Khan, and A. El Saddik, "SSIVD-Net: A Novel Salient Super Image Classification & Detection Technique for Weaponized Violence," *Lecture Notes in Networks and Systems*, vol. 1018 LNNS, pp. 16–35, Jul. 2022, doi: 10.1007/978-3-031-62269-4_2.
- [19] M. M. Mukto *et al.*, "Design of a real-time crime monitoring system using deep learning techniques," *Intelligent Systems with Applications*, vol. 21, Mar. 2024, doi: 10.1016/j.iswa.2023.200311.