Egyptian Informatics Journal 26 (2024) 100455

Contents lists available at ScienceDirect

Egyptian Informatics Journal

journal homepage: www.sciencedirect.com

Full Length Article

A shallow 3D convolutional neural network for violence detection in videos

Naz Dündar[a,*], Ali Seydi Keçeli[b], Aydın Kaya[b], Hayri Sever[c]

[a] *Department of Software Engineering, Çankaya University, Türkiye*
[b] *Department of Computer Engineering, Hacettepe University, Türkiye*
[c] *Department of Computer Engineering, Çankaya University, Türkiye*

ABSTRACT

With the recent worldwide statistical rise in the amount of public violence, automated violence detection in surveillance cameras has become a matter of high importance. This work introduces an end-to-end, trainable 3D Convolutional Neural Network (3D CNN) for detecting violence in video footage. The proposed network is inherently capable of processing both spatial and temporal information, thereby obviating the need for additional models that would introduce higher computational requirements and complexity. This work has two main contributions: 1) developing a lightweight 3D CNN suitable for inference on edge devices as mobile systems, and 2) a comprehensive explanation of all components comprising a CNN model, thereby enhances model interpretability. Experiments were conducted to assess the performance of the proposed model using a consolidated dataset combining four benchmark datasets. The results of the experiments support the asserted contributions, which are discussed in detail.

1. Introduction

In recent years, there has been a significant increase in the amount of public violence. This increase brought about a worldwide concern for preserving the security and safety of public places. Consequently, surveillance cameras have been installed in several places to continuously monitor public scenes. Despite the installment of surveillance cameras, there remains the problem of consistent manual human inspection to address disturbances or anomalies promptly. However, the human decision-making process is often characterized as slow and biased. This inhibition limits the capacity for simultaneous monitoring in the case of having multiple cameras. Moreover, employing humans for such tasks leads to a significant increase in costs.

The limitations of human supervision indicate the need for more reliable and effective monitoring system alternatives, especially in time-sensitive situations such as violence prevention. As a result, researchers have turned to computerized systems for solutions, with a specific focus on Machine Learning (ML), a subset of Artificial Intelligence (AI). ML involves training machines to learn patterns from data, which enables them to make predictions or decisions for similar problems. The integration of ML models into various domains and industries has significantly reduced human-dependency in many tasks and resulted in higher efficiency, performance, and efficiency.

The field of ML has advanced substantially over the years, especially with the emergence of Deep Learning (DL), particularly in pattern recognition tasks. Violence detection can be characterized as a pattern recognition problem, where machines can be taught to identify patterns in videos that indicate the presence of a violent act. This ability allows machines to identify violence quickly and unbiasedly, instantaneously alerting authorities to take the necessary actions to prevent such acts of violence. Moreover, machines can respond to multiple simultaneous stimulations, which immediately outperforms the capabilities of humans. Also, implementing automated violence detection systems would result in significant reductions in cost in the long run. These points support the consensus that ML models are more beneficial alternatives to manpower in many tasks, particularly violence detection in public places.

To that end, a violence detection framework is proposed in this work. In order to design a violence detection model, violence must be explicitly defined. In the context of this research, violence is defined as a voluntary, physical action exercised by one or more people with the intent of hurting one or more people against their will. While the extent of violence is not limited to this definition, this research focuses on detecting violent acts in alignment with the provided definition. The proposed model has a considerably small model size with a reduced number of parameters. Reducing parameters in a deep model has several advantages. It improves computational efficiency, leading to faster training and reduced memory requirements, making the model more suitable for resource-constrained environments like mobile devices. A smaller number of parameters often results in reduced overfitting and

* Corresponding author.
E-mail address: naz.dundar@cankaya.edu.tr (N. Dündar).

https://doi.org/10.1016/j.eij.2024.100455
Received 13 September 2023; Received in revised form 29 November 2023; Accepted 12 March 2024
Available online 21 March 2024

enhances model interpretability. Additionally, models with fewer parameters are well-suited for transfer learning and deployment on edge devices, contributing to energy efficiency. Striking a balance between model complexity and performance is crucial, as overly reducing parameters may compromise the model's representational capacity. The benefits of parameter reduction depend on the specific needs and constraints of the application at hand. Additionally, comparable results to more complex models proposed in the literature are obtained.

This paper is organized as follows. A thorough literature review of violence detection frameworks is presented in Section 2. The proposed model is explained in detail in Section 3. The settings and implementation details of the experiments, as well as the results, are given in Section 4. Final assessment and ideas for future research are presented in Section 5, thus concluding this paper.

2. Related works

This section presents previous works about violence detection, divided into two categories: Hand-Crafted and DL methods.

2.1. Hand-crafted methods

Early studies on violence detection in video mostly used visual or audio features to detect flame and blood [29], skin and blood [14], gunshots and explosions using Gaussian mixture models and Hidden Markov Models [13], etcetera. Later, the Bag-of-Words (BoW) procedure, often used for images, was adapted to videos [36] and was used frequently for video classification tasks. For example, [10] used spatiotemporal video cubes and the BoW approach for aggressive behavior detection. [30] developed a method for verifying person identity and detecting unusual human behavior based on the descriptors derived from Histograms of Optical Flow (HOF) at the automated Access Control Points (ACP). [7] used the BoW framework with action descriptors STIP [25] and MoSIFT [11]. They also created the Hockey Fights Dataset [7], one of the most commonly used datasets in violence detection tasks. [16] considered the change in the flow-vector magnitudes over time and used the Violent Flows (ViF) descriptor to represent these statistics, then classified ViF descriptors using linear Support Vector Machine (SVM). The Bag-of-Visual-Words (BoVW) approach was tested by [44] to compare optical flow algorithms such as Farnebäck [15], Horn-Schunck [18] and Lucas-Kanade [27]. They used descriptors based on local 3D volumes of Histograms of Oriented Gradients (HOG) and Histograms of Optical Flow (HOF). They used Random Forest [8] and the Fisher Kernel [21] for the final representation of the video. [49] used the MoSIFT algorithm to extract the low-level description of a video, Kernel Density Function (KDE) to eliminate the feature noise and sparse coding instead of BoW for human action representation. [50] used the Gaussian Model of Optical Flow (GMOF) to extract regions in the video that are candidates for violent activity. They proposed a novel descriptor, Orientation Histogram of Optical Flow (OHOF), which is given as an input to a linear SVM for the classification of the activity as violent or nonviolent. [6] proposed a ViF variation using Horn-Schunck as an optical flow algorithm and SVM for the classification of violent events. In another work, [6], they used the method proposed in [5] and, in addition, applied the non-adaptive interpolation super-resolution algorithm to improve the video quality and fire Kanade-Lucas-Tomasi (KLT) face detector to detect not only violence but also identify the person. [48] analyzed the features of the motion vectors in each frame and between the frames and got Region Motion Vectors descriptor (RMV), then used SVM for classification. [33] proposed a novel feature using Lagrangian direction fields based on a spatiotemporal model. They applied an extended BoW procedure for classification for each video to ensure proper spatial and temporal feature scales.

While the hand-crafted methods discussed above demonstrated satisfactory performance given the technology available then, many did not rely on GPUs for computations. Nevertheless, these methods were computationally intensive and were not able to achieve high accuracies. Moreover, their adaptability to diverse situations and environments was limited, making them less suitable for real-world applications. In response to the need for faster, less computationally expensive, more power-efficient models in violence detection, as well as other pattern recognition tasks, researchers turned to DL. This shift was aligned with the rapid technological advancements, and DL emerged as a promising approach to overcome the limitations associated with hand-crafted methods.

2.2. Deep learning methods

3D CNNs were used to extract spatiotemporal features and the output was fed to a multi-class linear SVM for classification in [43]. [40] used the pre-trained CNN model AlexNet [24] for feature extraction and aggregated the features using a Convolutional Long Short-Term Memory (ConvLSTM). [3] used CNN for spatial feature extraction and LSTM for classification. [45] proposed an end-to-end framework consisting of three steps. The first step is human detection using a lightweight CNN, the second is spatiotemporal feature extraction using a 3D CNN, and the third is the classification of the extracted features using a Softmax classifier. [37] used the VGG-16 [35] CNN model pre-trained on ImageNet [24] to extract spatial features followed by LSTM as the temporal feature extractor and a sequence of Fully-Connected layers for classification. A modified 3D CNN was presented by [38], where they improved the method for preprocessing the data and proposed a new sampling method by using the key frame as dividing nodes. [41] used three models: VGG-16 [35], VGG-19 [35] and ResNet50 [17], pre-trained on ImageNet [24] for feature extraction. They applied CNN, LSTM, and ConvLSTM to the dataset they created consisting of violent and nonviolent videos in different settings in the Bangladesh context. They also experimented with Spatial Transformer Network, a particular type of attention mechanism, and applied attention to the extracted features in some of their experiments. [4] combined a pre-trained 3D CNN with a linear SVM classifier to reduce overfitting and make a broader generalization. A novel violence detection framework that can be combined with 2D CNNs was proposed by [23]. They also proposed a spatial attention module called Motion Saliency Map (MSM) and a temporal attention module called Temporal Squeeze-and-Excitation (T-SE) to improve the model's performance. [20] presented a two-stream architecture leveraging Separable Convolutional LSTM (SepConvLSTM) and the pre-trained MobileNet [31] model. [28] proposed an approach that combines the VGG-16 [35] model pre-trained on ImageNet [24] with ConvLSTM [16]. [47] used ResNet50 [17] for feature extraction followed by ConvLSTM for detecting anomalies. [42] proposed a combination of 3D CNN and CNN Bidirectional LSTM (CNN-BiLSTM). [46] proposed two methods, 3D DenseNet Fusion OF RGB and 3D DenseNet Fusion OFnom RGB, and developed a new dataset called AICS-violence.

3. Proposed model

This section provides an overview of CNNs, describes their architecture, and introduces the proposed model.

2D CNNs exhibit considerable efficacy in static image recognition tasks. Therefore they are frequently used for such tasks. This success can be associated with 2D CNNs being specifically designed to handle spatial information present in images. However, there is strong spatial and temporal information inherent in videos. The limitation of 2D CNNs leads to the loss of motion information in videos, making them an insufficient alternative to handle video data.

In contrast, 3D CNNs incorporate a third dimension, *time*, enabling them to effectively capture spatiotemporal information. This distinct characteristic allows 3D CNNs to exploit motion information obtained from the differences between consecutive frames. Consequently, 3D CNNs appear as more suitable models for video recognition tasks. The objective of this work is to develop an end-to-end, trainable, lightweight

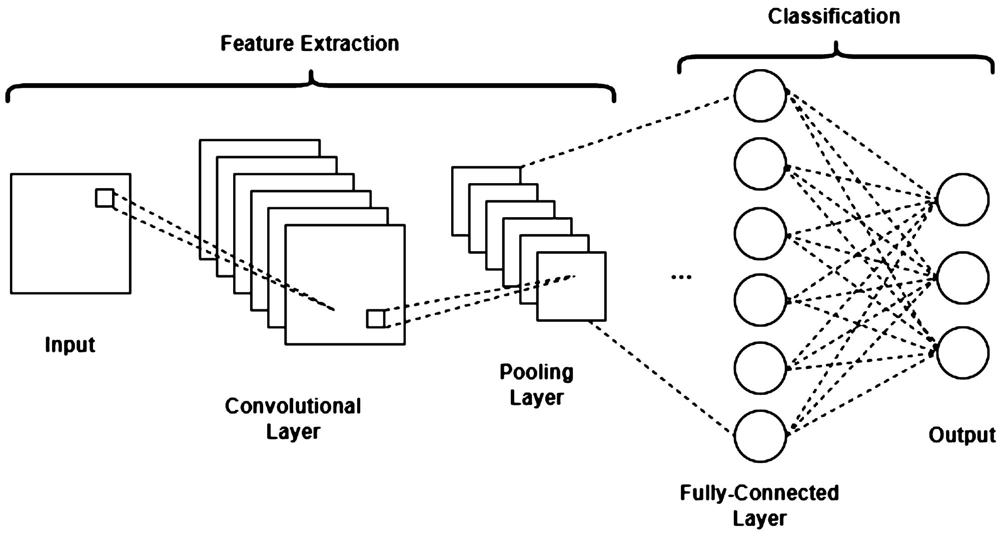


Fig. 1. The architecture of a typical CNN model.

3D CNN designed to extract spatiotemporal features and classify videos into violent and nonviolent categories. The benefits of a 3D CNN lie in its capacity to capture both spatial and temporal features. This is particularly advantageous in applications such as video analysis. Unlike 2D CNNs, 3D CNNs can model dynamic changes over time, making them well-suited for tasks where temporal information is crucial. The ability to consider the temporal dimension enhances the network's capability to recognize complex patterns and relationships within volumetric datasets. However, it's important to be mindful of the increased computational requirements associated with 3D CNNs and to consider their application in scenarios where spatiotemporal insights are essential for accurate analysis. Nonetheless, the main objective of this work was to design a 3D CNN model with a few parameters, allowing for inference in mobile systems with limited memory and scarce computational resources.

3.1. *Convolutional neural networks*

CNNs are a particular type of Deep Neural Network (DNN), distinguished by the incorporation of the convolution operation. They were first developed and introduced in the 1980s by Yann LeCun. CNN is a feed-forward neural network that uses the backpropagation algorithm for training, which is a type of supervised learning. They consist of multiple layers of artificial neurons, also referred to as nodes. These nodes are mathematical functions used to calculate the weighted sum of the inputs and give the output in the form of an activation map, highlighting significant features within the input. CNNs process their input represented by a pixel matrix, which is essentially a matrix of numbers. Each node within a CNN accepts input in the form of a matrix, calculates the product of their values and their corresponding assigned weights, adds them up, and channels them through an activation function.

A typical CNN architecture is comprised of three main types of layers: convolutional layers, pooling layers, and fully-connected layers stacked sequentially. The output of each layer serves as input for the subsequent layer. The configuration and characteristics of these layers can be adjusted, and additional layers may be included based on the task. CNNs perform in two phases: feature extraction, spanning from the input layer to the final pooling layer, and classification, which occurs in the fully-connected layer. The typical architecture of a CNN is given in Fig. 1.

The input of a CNN undergoes a series of floating point operations and matrix computations. Recent technological developments have facilitated an increase in the depth of CNNs, which refers to the number of layers they incorporate, to improve network performance. Consequently, this enhancement led to a notable increase in both computational complexity and size of CNNs. Although CNNs are rate-based neural networks, meaning they are suitable for CPU implementation, most new CNN models now require more powerful computing platforms, such as GPUs, due to the increased computational intensity [9]. For example, the ConvLSTM model proposed by [40] uses 9.6 million parameters and 14.40 billion floating-point operations. This scale of model size and computational intensity highlights the stronger computing resource requirements of modern CNNs.

3.2. *Model architecture*

In the proposed 3D CNN model, the input video undergoes 3D Convolution, 3D Max-Pooling, and Batch Normalization twice, a Flatten layer and a Fully-Connected (Dense) layer. Given that video data, such as AVI files, is used in the experiments, the input data is structured as a 5-dimensional object with dimensions [batch_size, number_of_frames, height, width, channels]. The layout of the proposed model is given in Fig. 2.

In a neural network, parameters refer to the weights and biases used for computation across all neurons. The proposed model incorporates a total of $54,634$ parameters, with $54,602$ being trainable, and 32 being non-trainable. While the overall parameter count is considerably low, keeping in mind that 3D CNNs process volumetric data, it should be noted that its parameter count does not solely determine the efficacy of a network. In fact, a reduced parameter count in 3D CNNs may hinder their ability to accurately model the complex relationships in the input data. Nonetheless, the main objective of this work was to design a 3D CNN model with a few parameters, allowing for inference in mobile devices with limited memory and scarce computational resources. Through several experiments, the parameter count has been successfully reduced without compromising a significant degree of accuracy.

3.2.1. *3D convolutional layer*

In Mathematics, convolution refers to the calculation of how the shape of a function is influenced by another function. It uses two functions as input and generates a third function as the output. In DL, 2D convolution refers to the operation of sliding a kernel (filter) of a specified size $M \times M$ with learnable weights across the input image, which is represented as a pixel matrix. The scalar (dot) product is calculated between the kernel and the parts of the input image based on the size of the kernel. Consequently, the output of a convolutional layer is the weighted sum of input and weights, also referred to as an activation

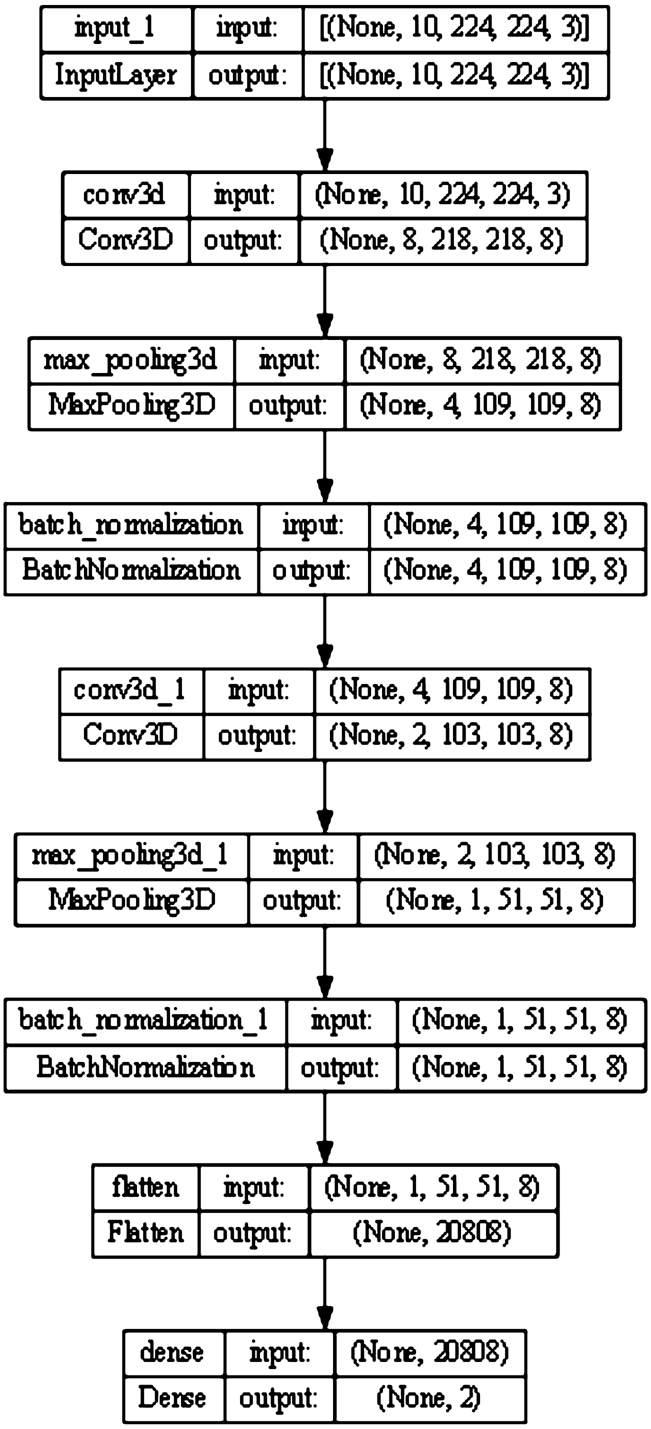

input_1 | input: | [(None, 10, 224, 224, 3)]
InputLayer | output: | [(None, 10, 224, 224, 3)]
conv3d | input: | (None, 10, 224, 224, 3)
Conv3D | output: | (None, 8, 218, 218, 8)
max_pooling3d | input: | (None, 8, 218, 218, 8)
MaxPooling3D | output: | (None, 4, 109, 109, 8)
batch_normalization | input: | (None, 4, 109, 109, 8)
BatchNormalization | output: | (None, 4, 109, 109, 8)
conv3d_1 | input: | (None, 4, 109, 109, 8)
Conv3D | output: | (None, 2, 103, 103, 8)
max_pooling3d_1 | input: | (None, 2, 103, 103, 8)
MaxPooling3D | output: | (None, 1, 51, 51, 8)
batch_normalization_1 | input: | (None, 1, 51, 51, 8)
BatchNormalization | output: | (None, 1, 51, 51, 8)
flatten | input: | (None, 1, 51, 51, 8)
Flatten | output: | (None, 20808)
dense | input: | (None, 20808)
Dense | output: | (None, 2)

Fig. 2. Layout.

map. In the context of images, the convolution operation requires the rotation of the kernel 180° counterclockwise, as a digital filter is used, which implies that the convolution is only applied after the feature vector is time-reversed [39]. The process of using the kernel without rotation is referred to as cross-correlation. In other words, convolution and cross-correlation are the same operation, differing only through kernel rotation. This distinction can be further explained mathematically, as presented in Equation (1).

$$f * g = f \star rot180°(g) \tag{1}$$

where f, g are functions, $rot180°(g)$ refers to the time-reversed form of g, $*$ represents convolution and $\star$ represents cross-correlation.

In this work, video data is used instead of image data. A video is a series of images shown consecutively to give the impression of continuous motion [38]. While an image has only spatial information, motion features between the adjacent frames of a video introduce a temporal dimension that must be considered. This added dimension makes 2D convolutions ineffective for robust feature extraction, since a substantial amount of temporal information is lost. To overcome this limitation, 3D convolutions are commonly used for feature extraction in video analysis tasks.

3D convolution operates similarly to 2D convolution, with the distinction that the kernel slides in three dimensions instead of two. 3D convolution is obtained using a 3D kernel on the cube formed by stacking adjacent frames together [4]. Consequently, movement information is acquired from adjacent frames. Thus, by using 3D convolution and 3D pooling, temporal information of the input video remains well preserved [38].

2D and 3D convolution operations are given in Equation (2) and Equation (3), respectively, where K is the convolution kernel, A is the convolution matrix, and B is the resulting matrix [26]. As shown in Equation (3), time dimension T is added to the 2D convolution given in Equation (2).

$$B(i,j) = \sum_{m=0}^{M}\sum_{n=0}^{N} K(m,n) * A(i-m, j-n) \tag{2}$$

$$B(i,j,r) = \sum_{m=0}^{M}\sum_{n=0}^{N}\sum_{t=0}^{T} K(m,n,t) * A(i-m, j-n, r-t) \tag{3}$$

In addition, [22] defined the formal equation for the value of a unit at position (x, y, z) in the jth feature map in the ith layer denoted by v_{ij}^{xyz}, given in Equation (4), where $\tanh(\cdot)$ is the hyperbolic tangent function, b_{ij} is the bias for this feature map, m indexes over the set of feature maps in the $(i-1)$th layer connected to the current feature map, w_{ijm}^{pqr} is the (p,q,r)th value of the kernel connected to the mth feature map in the previous layer, P_i and Q_i are the height and width of the kernel, respectively and R_i is the size of the 3D kernel along the temporal dimension. By this construction, the feature maps in the convolution layer are connected to multiple contiguous frames in the previous layer, thereby capturing motion information [22].

$$v_{ij}^{xyz} = \tanh\left(b_{ij} + \sum_{m}\sum_{p=0}^{P_i-1}\sum_{q=0}^{Q_i-1}\sum_{r=0}^{R_i-1} w_{ijm}^{pqr} v_{(i-1)m}^{(x+p)(y+q)(z+r)}\right) \tag{4}$$

In the proposed model, the 3D convolutional layers are configured with 8 filters, a kernel size of (3, 7, 7), and valid padding, implying that no zero-filled borders are added around the images. The stride is set to 1 by default, indicating the step size for each kernel movement, and the activation function applied is the Rectified Linear Unit (ReLU). The purpose of activation functions in neural networks is to transform a node's weighted sum into the node's activation. Various types of activation functions are commonly used, including ReLU, hyperbolic tangent, and sigmoid. ReLU is a piecewise function that retains the values of positive numbers and changes all negative numbers to zero. The equation of ReLU is given in Equation (5), where x is the input and $f(x)$ is the output of the function. ReLU is a default choice for many neural networks due to its facilitation of training and enhancement of network performance.

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{5}$$

3.2.2. 3D max-pooling layer

In most CNN architectures, each convolutional layer is typically followed by a pooling layer. The pooling operation, similar to convolution,

Table 1
Properties of the datasets used in the experiments.

Datasets	Resolution	Context	Number of violent videos	Number of nonviolent videos
Hockey Fights [7]	720 x 576	Hockey Games	500	500
Movies [7]	360 x 250	Movies	100	100
RWF-2000 [12]	Varying	Surveillance	1000	1000
RLVS [37]	480p–720p	Varying	1000	1000

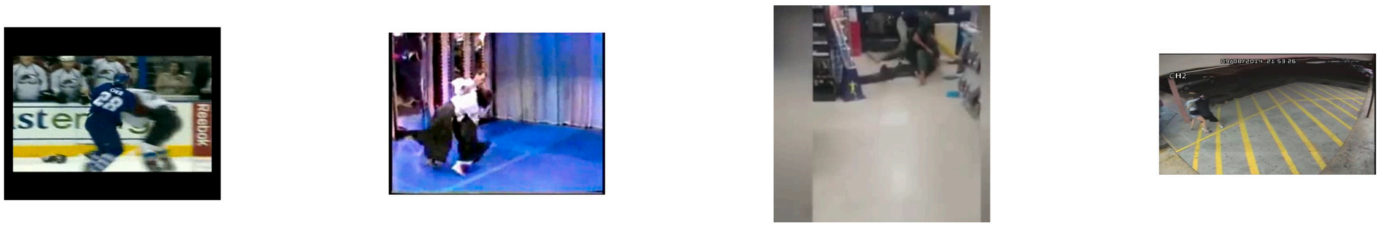


Fig. 3. Sample video frames of fights extracted from the datasets randomly. From left to right: Hockey fights, Movies, RLVS, RWF-2000.

groups up the pixels in the input image and filters them down to a subset. Pooling layers serve to decrease the size of the convolved feature map by decreasing the connections between layers. Various types of pooling exist, and the type used in this work is 3D Max-Pooling. The 3D Max-Pooling layer performs non-linear downsampling by dividing 3D input into cuboidal pooling regions and then computing the maximum of each region [1]. Although the term *downsampling* might be misconstrued as data loss, it is a crucial step in reducing overfitting and speeding up model training by denoising redundant data. In the proposed model, the 3D Max-Pooling layer kernel has a size of (2, 2, 2), and padding is valid.

3.2.3. *Batch normalization layer*

Normalization is commonly used to standardize raw data, thereby downscaling the data's range. This process increases the model's learning rate and convergence speed, preventing model divergence and consequently, facilitating training [32]. Batch normalization [19] is a particular type of normalization that differs from standard normalization by being applied between the model layers along training mini-batches. It can be added to the model as a layer. Batch normalization allows the use of higher learning rates, being less careful about initialization, and in some cases, eliminating the need for Dropout [19]. Given the use of batch normalization in the proposed model, the incorporation of Dropout was deemed unnecessary.

3.2.4. *Flatten layer*

As the name suggests, the flatten layer transforms the pooled feature maps, initially in the form of square matrices, into a one-dimensional column vector. Although the purpose of this layer is straightforward, it is an essential step in neural networks, given that neural networks accept input in the form of one-dimensional linear vectors. After the data is flattened, it becomes suitable for input into the fully-connected layer of the network.

3.2.5. *Dense layer*

The dense layer consists of neurons, each receiving input from all the neurons in the preceding layer, hence also referred to as fully-connected layer. The output of the convolutional layers is flattened in the preceding flatten layer to facilitate input into the dense layer. Following the extraction of features in the previous layers, the dense layer performs the classification based on the output of the convolutional layers. Given that the proposed model has two output classes, namely *fight* and *non-fight*, the shape of the final dense layer is set to 2. The Softmax function is the activation function used to normalize the input values into a probability distribution. Each input component, regardless of the value or sign, is converted to a number within the $(0,1)$ interval. The Softmax function is given in Equation (6), where $\sigma(z)$ denotes the Softmax function applied on vector z, K is the total number of elements in the vector, i, j are index variables representing the different elements in the vector, e is Euler's constant, and z_i, z_j are elements of the vector.

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{6}$$

4. Experiments

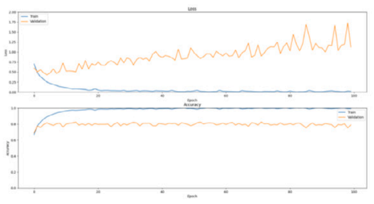
This section presents comprehensive details regarding the experiments, results, and comparisons with other works.

4.1. *Dataset descriptions*

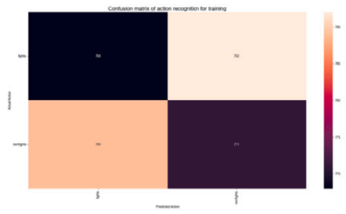
The violence detection datasets used in the experiments are Hockey Fights [7], Movies [7], RWF-2000 [12], and Real Life Violence Situations (RLVS) [37]. Table 1 provides the properties of these datasets, and Fig. 3 presents sample frames of fight scenes from each dataset. In the experiments, all four datasets were combined into one large dataset, consisting of 5200 videos in total, with 2600 labeled as violent and 2600 as nonviolent. This combined dataset was then divided into training, test, and validation sets, corresponding to 60%, 20%, and 20%, respectively. The percentage split corresponds to 3120 videos for training, 1040 for validation, and 1040 for the test set. Given that the model has two output classes, half of each set is violent videos, and the other half is nonviolent.

The training set is used to fit the model parameters based on the observational relationships between the data and their respective labels. Meanwhile, the validation set is used for hyperparameter tuning and approximating the model's predictive performance during training [32]. The split into training and validation sets is crucial to prevent overfitting, which is when the model gets too familiar with the training set and fails to generalize well to new, unseen data. The test set is not involved in the training process but shares the same predictive relationship as the training set and is used for evaluation.

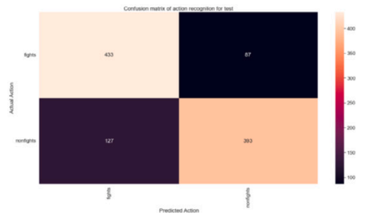
Splitting the videos into training, validation and test sets was done randomly in the algorithm for each experiment. This randomized approach was chosen to ensure that the model is trained and tested with a different selection of videos for each experiment, resulting in a more precise and accurate evaluation of the proposed model.



(a) Loss and Accuracy Graphs

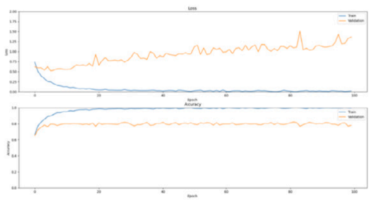


(b) Confusion matrix for the training set

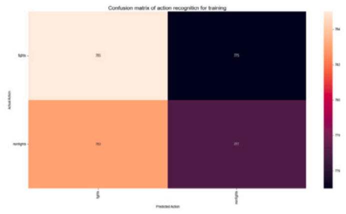


(c) Confusion matrix for the test set

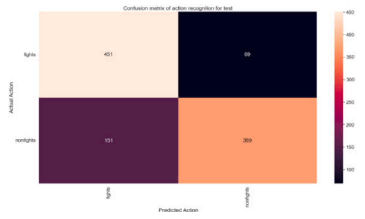
Fig. 4. Experiment 1.



(a) Loss and Accuracy Graphs

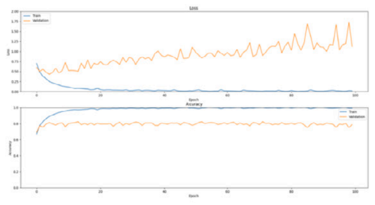


(b) Confusion matrix for the training set

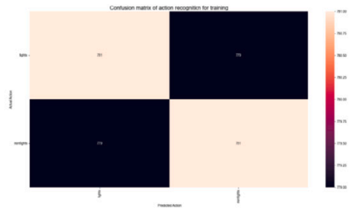


(c) Confusion matrix for the test set

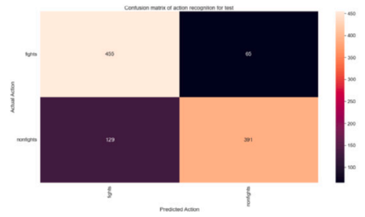
Fig. 5. Experiment 2.



(a) Loss and Accuracy Graphs



(b) Confusion matrix for the training set



(c) Confusion matrix for the test set

Fig. 6. Experiment 3.

4.2. Implementation details

All experiments were carried out on a computer equipped with 11th Gen Intel(R) Core(TM) i7 64GB RAM and 16 cores complemented by an NVIDIA GeForce GTX 1660 SUPER GPU. The implementation was done using Python 3.7, leveraging the Tensorflow framework, with the network layers imported from the Keras library. While the code's structure was inspired by [2], it was modified and adapted accordingly to the specifics of the proposed model.

4.3. Hyperparameters

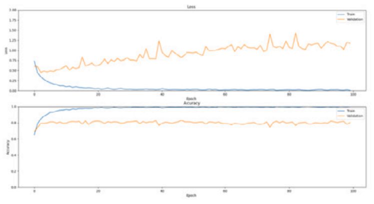
Hyperparameters play a crucial role in determining the network's structure and how it is trained. They are set before the training process begins. Among these hyperparameters, the learning rate is particularly significant in neural network training. If the learning rate is too large, the optimization process may diverge or miss potential local minima. Conversely, if the learning rate is too low, the training duration may be extended, as the convergence to a minimum would take longer compared to a higher learning rate [32]. Through experimentation, it was determined that a learning rate of 10^{-4} yielded optimal results for the proposed model.

In neural networks, the optimizer and the loss function work together to optimize the network's performance. A loss value is computed in each epoch, and the optimizer tries to compensate for the loss, thereby improving the model's overall performance. In the proposed model, training was conducted using the Adam algorithm as the optimizer and Sparse Categorical Cross Entropy as the loss function, which is a common choice for multiclass classification tasks. The batch size was set to 8, meaning that 8 videos were processed simultaneously in each iteration. After experimenting with different numbers of epochs, it was observed that the loss ceased to improve after 100 epochs. Consequently, the number of epochs was fixed at 100. Each epoch's average duration was approximately 640 seconds, corresponding to a total run-time of 17.7 hours. This run-time is expected for 3D CNNs, especially considering the dataset's substantial size and the moderate computational power of the GPU used in this work.

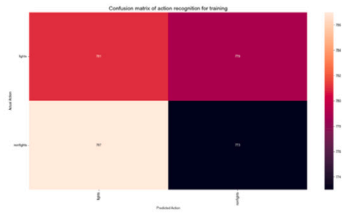
4.4. Results

Loss and accuracy graphs were generated for each experiment. The loss graph exhibits a decreasing trend in loss with each epoch, while the accuracy graph exhibits an increasing trend. Given the inherent relationship where accuracy naturally increases as loss decreases, the graphs mirror each other. Additionally, confusion matrices were created for both the training and test sets in each experiment. Confusion matrices contain the values of false positives, true positives, false negatives, and true negatives, providing a simple yet effective evaluation of the model's performance. The graphs and confusion matrices of all five experiments are presented in Figs. 4, 5, 6, 7 and 8.

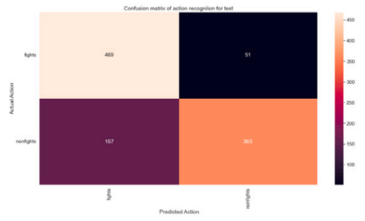
Five experiments were carried out on the combined dataset, and the performance of the proposed model was assessed using metrics such as



(a) Loss and Accuracy Graphs

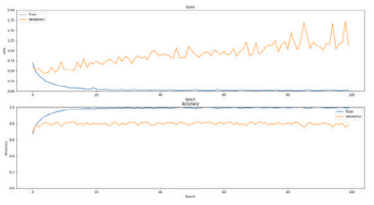


(b) Confusion matrix for the training set

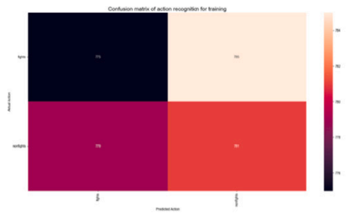


(c) Confusion matrix for the test set

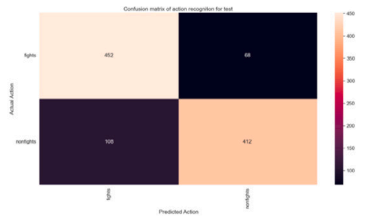
Fig. 7. Experiment 4.



(a) Loss and Accuracy Graphs



(b) Confusion matrix for the training set



(c) Confusion matrix for the test set

Fig. 8. Experiment 5.

Table 2
Accuracy, loss, precision, recall and F1-score values of each experiment for the proposed 3D CNN model.

Experiment	Accuracy	Loss	Precision	Recall	F1-Score
1	0.7981	1.3180	0.773214	0.832692	0.801852
2	0.7952	1.4102	0.749169	0.867308	0.803922
3	0.8135	1.2759	0.779110	0.875000	0.824275
4	0.7933	1.2523	0.749201	0.901923	0.818450
5	0.8288	1.2670	0.807143	0.869231	0.837037

Table 3
Overall accuracy, loss, precision, recall and F1-score values calculated as the average of all five experiments.

Accuracy	Loss	Precision	Recall	F1-Score
0.8058	1.3047	0.771567	0.869231	0.817107

accuracy, precision, recall, and F1-score. These metrics were computed based on the confusion matrices created from the test set of each experiment. The formulas of the metrics are given in Equations (7), (8), (9), and (10), where TP, TN, FP, FN denote true positive, true negative, false positive, and false negative, respectively. Their corresponding elements in the confusion matrices are the top left corner for TP, the top right corner for FN, the bottom left corner for FP and the bottom right corner for TN. Accuracy, loss, precision, recall, and F1-score values are presented in Table 2 for each experiment. The average of each evaluation metric of the five experiments was computed, which is considered to be the overall performance of the network. The overall accuracy, loss, precision, recall, and F1-score values are presented in Table 3.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{7}$$

$$Precision = \frac{TP}{TP+FP} \tag{8}$$

$$Recall = \frac{TP}{TP+FN} \tag{9}$$

$$F1-Score = \frac{2 \times precision \times recall}{precision + recall} \tag{10}$$

The overall accuracy computed for the proposed model was 80.58%, which is in alignment with expectations. Notably, the network was trained with 3120 out of the 5200 videos in the combined dataset, and many of these videos have low resolutions, with some featuring out-of-focus camera actions. Additionally, it should be acknowledged that accuracy was, to some extent, sacrificed to reduce the number of network parameters, enabling it to run efficiently on mobile systems, which was the main objective of this work.

While the run-time of this model with the combined dataset was 17.7 hours, the model is expected to achieve high performance on mobile devices once the training is completed. It is also expected that training the network with a larger number of videos of higher resolutions would increase the model's accuracy. Moreover, the accuracy of a neural network is also dependent on the computational power of the GPU used for the experiments. Thus, a more powerful GPU would not only decrease the run-time of the network but also improve its accuracy.

Precision is an assessment of the accuracy of the model's positive predictions, measuring its ability to correctly identify positive samples (i.e., fight videos in this context) while minimizing false positives. A precision value of 77.1567% indicates that the network has a relatively high rate of true positive predictions compared to false positives. This value suggests that the model consistently makes accurate positive predictions, which is the desired outcome.

Recall quantifies the proportion of true positive predictions out of all actual positive samples. A recall value of 86.9231% is an indication of the model's ability to correctly identify around 86.9231% of the actual positive samples in the dataset. This result suggests that the model has a considerably low rate of false negatives and can capture a significant number of positive instances.

Additionally, the F1-score metric represents the harmonic mean of precision and recall, providing a balanced evaluation of the model's performance. An F1-score of 81.7107% suggests that the model has achieved a good balance between precision and recall. Overall, these metrics allow for a comprehensive evaluation of the model's effectiveness in violence detection.

Numerous violence detection models have been published and evaluated on the four datasets used in this work. However, these evaluations were typically conducted on individual datasets. Since these datasets were combined into one large dataset, making direct, fair compar-

Table 4
Accuracy comparisons with other models.

Model	Year	Hockey	Movies	RWF	RLVS
MoSIFT+HIK [7]	2011	90.9%	89.5%	-	-
C3D [45]	2019	96%	99.9%	-	-
VGG16+LSTM [37]	2019	95.1%	99%	-	71.5%
C3D+SVM [4]	2020	98.51%	-	-	-
VGG16+ConvLSTM [28]	2021	99.1%	100%	92.4%	-
Xception+LSTM [34]	2021	96.5%	98.3%	-	-
SepConvLSTM-M [20]	2021	99.5%	100%	89.75%	-
CNN+BiLSTM [42]	2022	94.9%	92.9%	-	-
Proposed	2023	95.5%	100%	73%	88%

isons with previously proposed models was challenging. To address this limitation, five experiments were conducted for each dataset, and the average accuracy across these experiments was computed, presenting a fair basis for comparison with other works in the literature. The results of the individual experiments and the accuracy values of some other works in literature, are presented in Table 4.

While the proposed model did not outperform other works in the literature, it performed well within the expected range. Moreover, the primary objective was to create a model substantially smaller in size than other existing models to enable mobile inference, which was achieved. Consequently, the proposed model is substantially smaller in size than most other works in the literature, resulting in a reduction in computational requirements. In addition, since the four datasets were combined into one large dataset, the model was trained on a much larger dataset than other proposed models, making it more capable of identifying violence in unseen data. Training the model with a larger dataset of varying context and resolutions also significantly reduces overfitting, resulting in a more consistent evaluation of the model's performance. The performance of the proposed model can be further improved to achieve higher results, which is a consideration for future works.

5. Conclusion

This paper introduces an end-to-end, trainable 3D CNN designed to detect physical fights in videos. The model processes video input, extracts features, and performs classification, all within a single feed-forward 3D CNN architecture. The model's evaluation uses a combined dataset of four publicly available datasets: Hockey Fights, Movies, RLVS, and RWF-2000. Five experiments were conducted on this combined dataset, with the training, test, and validation sets randomly selected by the network at the beginning of each experiment.

Real-time violence detection poses a considerable challenge. To integrate a violence detection framework seamlessly into a real-life surveillance system, ensuring it does not overlook anomalies or trigger false alarms, the model must undergo training on a substantial number of videos. Training a model on a larger dataset introduces diverse situations and contexts, leading to higher accuracy. However, processing large amounts of volumetric data significantly increases the network's run-time, memory usage, and storage requirements. Given the available resources and data, along with the computational efficiency of the proposed model, a satisfactory trade-off between accuracy and model size was achieved.

While all evaluation metrics scored above the upper quartile, the proposed model, designed to detect violent activity in public places, requires further improvement to be used as the sole detector of a delicate and time-sensitive matter such as violence. The main objective of this paper was to design a lightweight CNN that is considerably small in size to detect violence, which was achieved. The model's accuracy and performance can be improved through hyperparameter tuning, weight initialization, and the exploration of efficient combinations with other ML algorithms. These improvements are aimed to be considered in future works, as well as extending the definition of violence beyond physical fights, ensuring comprehensive coverage of all types of violence and threats to society.

CRediT authorship contribution statement

Naz Dündar: Methodology, software, writing. **Ali Seydi Keçeli:** Idea development, software, editing. **Aydın Kaya:** Investigation, data collection, editing. **Hayri Sever:** Idea development, supervision, reviewing.

Declaration of competing interest

None.

References

[1] maxpooling3dlayer. https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.maxpooling3dlayer.html. [Accessed 30 January 2023].
[2] Video classification with a 3d convolutional neural network. https://www.tensorflow.org/tutorials/video/video_classification. [Accessed 20 June 2022].
[3] Abdali Al-Maamoon R, Al-Tuma Rana F. Robust real-time violence detection in video using cnn and lstm. In: 2019 2nd scientific conference of computer sciences (SCCS). IEEE; 2019. p. 104–8.
[4] Accattoli Simone, Sernani Paolo, Falcionelli Nicola, Mekuria Dagmawi Neway, Franco Dragoni Aldo. Violence detection in videos by combining 3d convolutional neural networks and support vector machines. Appl Artif Intell 2020;34(4):329–44.
[5] Machaca Arceda V, Fernández Fabián K, Gutíerrez Juan Carlos. Real time violence detection in video; 2016.
[6] Machaca Arceda VE, Fernández Fabián KM, Laguna Laura PC, Rivera Tito JJ, Gutiérrez Cáceres JC. Fast face detection in violent video scenes. Electron Notes Theor Comput Sci 2016;329:5–26.
[7] Nievas Enrique Bermejo, Suarez Oscar Deniz, García Gloria Bueno, Sukthankar Rahul. Violence detection in video using computer vision techniques. In: International conference on computer analysis of images and patterns. Springer; 2011. p. 332–9.
[8] Breiman Leo. Random forests. Mach Learn 2001;45:5–32.
[9] Cao Yongqiang, Chen Yang, Khosla Deepak. Spiking deep convolutional neural networks for energy-efficient object recognition. Int J Comput Vis 2015;113(1):54–66.
[10] Chen Datong, Wactlar Howard, Chen Ming-yu, Gao Can, Bharucha Ashok, Hauptmann Alex. Recognition of aggressive human behavior using binary local motion descriptors. In: 2008 30th annual international conference of the IEEE engineering in medicine and biology society. IEEE; 2008. p. 5238–41.
[11] Chen Ming-yu, Hauptmann Alexander. Mosift: recognizing human actions in surveillance videos; 2009.
[12] Cheng Ming, Cai Kunjing, Li Ming. Rwf-2000: an open large scale video database for violence detection. In: 2020 25th international conference on pattern recognition (ICPR). IEEE; 2021. p. 4183–90.
[13] Cheng Wen-Huang, Chu Wei-Ta, Wu Ja-Ling. Semantic context detection based on hierarchical audio models. In: Proceedings of the 5th ACM SIGMM international workshop on multimedia information retrieval; 2003. p. 109–15.
[14] Clarin Christine, Dionisio J, Echavez Michael, Dove Prospero Naval. Detection of movie violence using motion intensity analysis on skin and blood. PCSC 2005;6:150–6.
[15] Farnebäck Gunnar. Two-frame motion estimation based on polynomial expansion. In: Image analysis: 13th Scandinavian conference, SCIA 2003 Halmstad, Sweden, June 29–July 2, 2003 proceedings 13. Springer; 2003. p. 363–70.
[16] Hassner Tal, Itcher Yossi, Kliper-Gross Orit. Violent flows: real-time detection of violent crowd behavior. In: 2012 IEEE computer society conference on computer vision and pattern recognition workshops. IEEE; 2012. p. 1–6.
[17] He Kaiming, Zhang Xiangyu, Ren Shaoqing, Sun Jian. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2016. p. 770–8.
[18] Horn Berthold KP, Schunck Brian G. Determining optical flow. Artif Intell 1981;17(1–3):185–203.
[19] Ioffe Sergey, Szegedy Christian. Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning; 2015. p. 448–56.pmlr.
[20] Islam Zahidul, Rukonuzzaman Mohammad, Ahmed Raiyan, Kabir Md Hasanul, Farazi Moshiur. Efficient two-stream network for violence detection using separable convolutional lstm. In: 2021 international joint conference on neural networks (IJCNN). IEEE; 2021. p. 1–8.
[21] Jaakkola TS, Haussler D. Exploiting generative models in discriminative classifiers. In: Kearns MS, Solla SA, Cohn DA, editors. Advances in neural information processing systems, vol. 11. 1999.
[22] Ji Shuiwang, Xu Wei, Yang Ming, Yu Kai. 3d convolutional neural networks for human action recognition. IEEE Trans Pattern Anal Mach Intell 2012;35(1):221–31.

[23] Kang Min-seok, Park Rae-Hong, Park Hyung-Min. Efficient spatio-temporal modeling methods for real-time violence recognition. IEEE Access 2021;9:76270–85.
[24] Krizhevsky Alex, Sutskever Ilya, Hinton Geoffrey E. Imagenet classification with deep convolutional neural networks. Commun ACM 2017;60(6):84–90.
[25] Laptev Ivan. On space-time interest points. Int J Comput Vis 2005;64(2):107–23.
[26] Li Chengyang, Zhu Liping, Zhu Dandan, Chen Jiale, Pan Zhanghui, Li Xue, et al. End-to-end multiplayer violence detection based on deep 3d cnn. In: Proceedings of the 2018 VII international conference on network, communication and computing; 2018. p. 227–30.
[27] Lucas Bruce D, Kanade Takeo. An iterative image registration technique with an application to stereo vision. In: IJCAI'81: 7th international joint conference on artificial intelligence, vol. 2. 1981. p. 674–9.
[28] Mugunga Israel, Dong Junyu, Rigall Eric, Guo Shaoxiang, Madessa Amanuel Hirpa, Nawaz Hafiza Sadia. A frame-based feature model for violence detection from surveillance cameras using convlstm network. In: 2021 6th international conference on image, vision and computing (ICIVC). IEEE; 2021. p. 55–60.
[29] Nam Jeho, Alghoniemy Masoud, Tewfik Ahmed H. Audio-visual content-based violent scene characterization. Proceedings 1998 international conference on image processing. ICIP98 (Cat. No. 98CB36269), vol. 1. IEEE; 1998. p. 353–7.
[30] Perš Janez, Sulić Vildana, Kristan Matej, Perše Matej, Polanec Klemen, Kovačič Stanislav. Histograms of optical flow for efficient representation of body motion. Pattern Recognit Lett 2010;31(11):1369–76.
[31] Sandler Mark, Howard Andrew, Zhu Menglong, Zhmoginov Andrey, Chen Liang-Chieh. Mobilenetv2: inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2018. p. 4510–20.
[32] Schilling Fabian. The effect of batch normalization on deep convolutional neural networks; 2016.
[33] Senst Tobias, Eiselein Volker, Kuhn Alexander, Sikora Thomas. Crowd violence detection using global motion-compensated Lagrangian features and scale-sensitive video-level representation. IEEE Trans Inf Forensics Secur 2017;12(12):2945–56.
[34] Sharma Sarthak, Sudharsan B, Naraharisetti Saamaja, Trehan Vimarsh, Jayavel Kayalvizhi. A fully integrated violence detection system using cnn and lstm. Int J Electr Comput Eng 2021;11(4).
[35] Simonyan Karen, Zisserman Andrew. Very deep convolutional networks for large-scale image recognition. Preprint. arXiv:1409.1556, 2014.
[36] Sivic Josef, Zisserman Andrew. Video Google: a text retrieval approach to object matching in videos. IEEE international conference on computer vision, vol. 3. IEEE Computer Society; 2003. p. 1470.
[37] Soliman Mohamed Mostafa, Kamal Mohamed Hussein, Abd El-Massih Nashed Mina, Mohamed Mostafa Youssef, Chawky Bassel Safwat, Khattab Dina. Violence recognition from videos using deep learning techniques. In: 2019 ninth international conference on intelligent computing and information systems (ICICIS). IEEE; 2019. p. 80–5.
[38] Song Wei, Zhang Dongliang, Zhao Xiaobing, Yu Jing, Zheng Rui, Wang Antai. A novel violent video detection scheme based on modified 3d convolutional neural networks. IEEE Access 2019;7:39172–9.
[39] Stankovic Ljubisa, Mandic Danilo. Convolutional neural networks demystified: a matched filtering perspective based tutorial. Preprint. arXiv:2108.11663, 2021.
[40] Sudhakaran Swathikiran, Lanz Oswald. Learning to detect violent videos using convolutional long short-term memory. In: 2017 14th IEEE international conference on advanced video and signal based surveillance (AVSS). IEEE; 2017. p. 1–6.
[41] Sumon Shakil Ahmed, Goni Raihan, Hashem Niyaz Bin, Shahria Tanzil, Rahman Rashedur M. Violence detection by pretrained modules with different deep learning approaches. Vietnam J Comput Sci 2020;7(01):19–40.
[42] Talha Khalid Raihan, Bandapadya Koushik, Monirujjaman Khan Mohammad. Violence detection using computer vision approaches. In: 2022 IEEE world AI IoT congress (AIIoT). IEEE; 2022. p. 544–50.
[43] Tran Du, Bourdev Lubomir, Fergus Rob, Torresani Lorenzo, Paluri Manohar. Learning spatiotemporal features with 3d convolutional networks. In: Proceedings of the IEEE international conference on computer vision; 2015. p. 4489–97.
[44] Uijlings Jasper RR, Duta Ionut Cosmin, Rostamzadeh Negar, Sebe Nicu. Realtime video classification using dense hof/hog. In: Proceedings of international conference on multimedia retrieval; 2014. p. 145–52.
[45] Ullah Fath U Min, Ullah Amin, Muhammad Khan, Haq Ijaz Ul, Baik Sung Wook. Violence detection using spatiotemporal features with 3d convolutional neural network. Sensors 2019;19(11):2472.
[46] Vo-Le Cuong, Vo Hung Sy, Vu Thien Duy, Son Nguyen Hong. Violence detection using feature fusion of optical flow and 3d cnn on aics-violence dataset. In: 2022 IEEE ninth international conference on communications and electronics (ICCE). IEEE; 2022. p. 395–9.
[47] Vosta Soheil, Yow Kin-Choong. A cnn-rnn combined structure for real-world violence detection in surveillance cameras. Appl Sci 2022;12(3):1021.
[48] Xie Jianbin, Yan Wei, Mu Chundi, Liu Tong, Li Peiqin, Yan Shuicheng. Recognizing violent activity without decoding video streams. Optik 2016;127(2):795–801.
[49] Xu Long, Gong Chen, Yang Jie, Wu Qiang, Yao Lixiu. Violent video detection based on mosift feature and sparse coding. In: 2014 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE; 2014. p. 3538–42.
[50] Zhang Tao, Yang Zhijie, Jia Wenjing, Yang Baoqing, Yang Jie, He Xiangjian. A new method for violence detection in surveillance scenes. Multimed Tools Appl 2016;75(12):7327–49.