This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3155123, IEEE Access

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Low-cost CNN for Automatic Violence Recognition on Embedded System

JOELTON CEZAR VIEIRA[1], ANDREZA SARTORI[1,2], STÉFANO FRIZZO STEFENON[3,4], FÁBIO LUIS PEREZ[1], GABRIEL SCHNEIDER DE JESUS[2], and VALDERI REIS QUIETINHO LEITHARDT[5,6] (Member, IEEE)

[1]Department of Telecom., Electrical and Mechanical Engineering, Regional University of Blumenau (FURB). Rua São Paulo 3250, 89030-000 Blumenau, Brazil
[2]Department of Information Systems and Computing, Regional University of Blumenau (FURB). Rua Antônio da Veiga 140, 89030-903 Blumenau, Brazil
[3]Fondazione Bruno Kessler. Via Sommarive 18, 38123 Povo, Trento, Italy
[4]Computer Science and Artificial Intelligence, University of Udine. Via delle Scienze 206, 33100 Udine, Italy
[5]COPELABS, Lusófona University of Humanities and Technologies, Campo Grande 376, 1749-024 Lisboa, Portugal
[6]VALORIZA, Research Center for Endogenous Resources Valorization, Instituto Politécnico de Portalegre, 7300-555 Portalegre, Portugal

Corresponding author: Andreza Sartori (e-mail: asartori@furb.br).

This work was supported by national funds through the Fundação para a Ciência e a Tecnologia, I.P. (Portuguese Foundation for Science and Technology) by the project UIDB/05064/2020 (VALORIZA—Research Centre for Endogenous Resource Valorization), and Project UIDB/04111/2020, ILIND–Instituto Lusófono de Investigação e Desenvolvimento, under project COFAC/ILIND/COPELABS/3/2020.

ABSTRACT Due to the increasing number of violence cases, there is a high demand for efficient monitoring systems, however, these systems can be susceptible to failure. Therefore, this work proposes the analysis and application of low-cost Convolutional Neural Networks (CNNs) techniques to automatically recognize and classify suspicious events. Thus, it is possible to alert and assist the monitoring process with a reduced deployment cost. For this purpose, a dataset with violence and non-violence actions in scenes of crowded and non-crowded environments was assembled. The mobile CNNs architectures were adapted and obtained a classification accuracy of up to 92.05%, with a low number of parameters. To demonstrate the models' validity, a prototype was developed by using an embedded Raspberry Pi platform, able to execute a model in real-time with 4 frames-per-second of speed. In addition, a warning system was developed to recognize pre-fight behavior and anticipate violent acts, alerting security to potential situations.

I. INTRODUCTION

There is a growing interest in intelligent surveillance systems due to major concerns about global security and the need for effective monitoring of public places, such as urban centers, airports, railway stations, malls, sports stadiums, tourist venues, etc. Indeed, the number of cameras installed in urban centers and public places is increasing progressively, in order to promote order and safety [1].

Currently, most public monitoring systems are performed through security cameras, mainly in areas with a large flow of people. In fact, monitoring systems are basically composed of several cameras positioned at strategic locations. The problem with this systems consist of having only one human agent responsible for tracking the video input from many cameras simultaneously [2]. This can lead to errors when identifying suspicious events, often caused by inattention or fatigue. As such, monitoring systems are used as a storage system for possible lawsuits, rather than an incident prevention system as it should be [3].

In an automatic surveillance system, computers continually process a video input and scrutinize each frame to find a suspicious event, in which they can immediately report to the supervisor for their attention [4]. For this reason, the use of Computer Vision and Machine Learning techniques applied to automatic recognition of violence can help monitoring agents significantly.

The research in Computer Vision, specifically in action recognition, has mainly focused on detecting simple actions, such as walking or sports activities. The detection and recognition of fights or aggressive behavior in general has been comparatively less studied [2]. In practice, the automatic recognition of aggressive and irregular actions can be extremely useful for various video surveillance scenarios outside urban centers, such as in prisons, psychiatric centers, and even in monitoring daily activities and detecting falls of elderly people in homes [5]. Neural network models are increasingly being applied in embedded systems [6]–[10] to improve the ability to analyze data from different equipment

[11]–[14].

Thus, this work proposes an automatic violence recognition system in real time, able to distinguish between acts of violence, such as fights, vandalism, looting and shooting with firearms, and non-violence actions, such as walking, running, hugging, kissing, exercising, and celebrating. For this, several videos were assembled from a diverse range of public datasets, containing varied lighting conditions, scale, movement, number of people and objects. Moreover, the study used and compared different models of Mobile Convolutional Neural Networks. To avoid overfitting and improve the model generalization, traditional data augmentation techniques were applied. Then, the models were implemented and evaluated on a Raspberry Pi 4 embedded platform.

To summarize, the contributions of this work are the following:

- A quantitative and qualitative analysis of state-of-the-art Mobile CNN architectures for the binary recognition of violence actions;
- A dataset for the violence acts recognition was set up, through a manual selection and combination of public datasets. This dataset has several human actions and activities divided into two classes: violence and non-violence. In addition, contains scenes of violence in crowded and non-crowded environments;
- It was possible to achieve an accuracy of up to 92.05%, with models containing 2.26 million parameters;
- The trained models are able to recognize various actions of violence, such as punching, kicking, fighting, attacking, destroying, aiming and firing guns, wresting, and boxing.
- A prototype of a low-cost, intelligent monitoring system was developed on a Raspberry Pi embedded platform, able to run a mobile CNN model with a processing speed of up to 4 frames-per-second.
- A novel approach was created with the warning system, which is able to recognize pre-fight behavior and alert security to take appropriate action.

This paper is structured as follows: Section 2 presents the related work, with a brief description of existing automatic violence recognition works and the methods used. Section 3 corresponds to the development of the dataset, the selected public datasets, the distribution of training and testing set and a video duration histogram. In Section 4 is described the adapted versions of the mobile CNNs with the prepossessing steps. Section 5 presents the experiments, with a comparison of the developed models results, analysing the error, accuracy and number of parameters. Section 6 details the development, operation and comparison of the prototype. Section 7 presents a novel approach with the warning system. The last section is devoted to the final discussion and conclusions.

II. RELATED WORK

Recently, Deep Learning techniques, such as Convolutional Neural Networks (CNN or ConvNet), have shown excellent results in image and video classification [15]–[18]. In different challenges and datasets, these structures have been performing much better than previous proposals [3]. In fact, there are three main advantages of using CNN models in intelligent monitoring systems. First, they are less affected by noise in the data. Second, they achieve higher accuracy than other methods, even sometimes greater than the human eye. Lastly, they have the ability to classify people into different orientations and postures. Moreover, they also do not required hand-crafted extractor for encoding features [19], as was performed before the introduction of Deep Learning [20].

For example, [21] developed a violence detection system in movie scene, in which applied various elements of Deep Learning. First a CNN was trained, then a two-stream CNN was used to extract both static and optical flow motion features. Finally, a Long Short Term Memory (LSTM) [22] was applied to extract the long term temporal features [23]. Complementary motion and audio information was also extracted.

Also, [24] presented a model using 3D Convolutional Neural Networks directly on raw input data that automatically extracts the features. The 3D ConvNets extracts static features such as traditional CNN 2D, as well as adds the temporal dimension, thus allowing the extraction of motion features. Model evaluation is performed on the Hockey dataset [2]. Similarly, [20] proposed a Deep Learning model based on 3D CNN, adapting bottleneck units and the DenseNet architecture to promote efficient extraction of spatiotemporal features. Currently, [25] proposes a method that extracts the video spatiotemporal features through a convolutional neural network and combines them with the trajectory features in order to detect violence in video.

While the aforementioned works successfully recognize and classify acts of violence with good accuracy, all require significant processing power, as usually are employing high-end systems with multiple GPUs or TPUs [26].

The application of Deep Neural Networks in embedded systems or with limited computational capacity has been considerably less studied. Applications such as face recognition, gender detection [27], and emotion recognition [28] are some of the real-time models developed implementing the Raspberry Pi embedded platform. However, no applications or models were found on the violence classification and recognition by only utilizing the embedded systems capabilities.

Although typically the training phase requires more processing power than running the model [29], computationally limited systems such as small computers and embedded systems still face major difficulties in reproducing such models efficiently [30]. For this reason, this work conducts an extensive evaluation of Deep Neural Networks recently developed for embedded platforms and mobile applications, namely mobile CNN architectures [31]. With this study, it was developed a low-cost intelligent monitoring system on a Raspberry Pi embedded platform able to recognize pre-fight behavior and alert security to take appropriate action.

III. DATASET

To successfully train any deep CNN model, many video samples are needed [32]. However, there are only a few public video datasets available, specifically for violence recognition. In addition, they have an insufficient number of videos or are not applicable for this work. For this reason, action recognition videos were gathered in public datasets, in which were manually selected the classes and videos to be used.

The dataset developed is divided into training and testing set with two classes: violence and nonviolence. The violence class contains violent actions, such as punching, kicking, fighting, attacking, destroying, aiming and firing guns, wresting, and boxing. The nonviolence class contains typical actions for target places, such as malls, airports, subways and public parks, actions such as walking, jogging, running, sitting, hugging, kissing, walking the dog, exercising, biking and, celebrating. Thus, for this work, four public datasets were selected and combined: Violent-Flow[1] [33], UCF-101[2] [34], HMDB[3] [35], and Moments in Time[4] [36].

The Violent-Flow dataset [33] is the only selected dataset exclusively for violence recognition. It is a real-world dataset, that is, recorded videos are real acts of violence, mostly in crowds, taken by surveillance cameras or smartphones. The dataset contains 246 labeled training videos and 44 test videos. Figure 1 shows some samples of violence and nonviolence from the dataset. It is possible to see that, indeed, most actions are from crowded places.

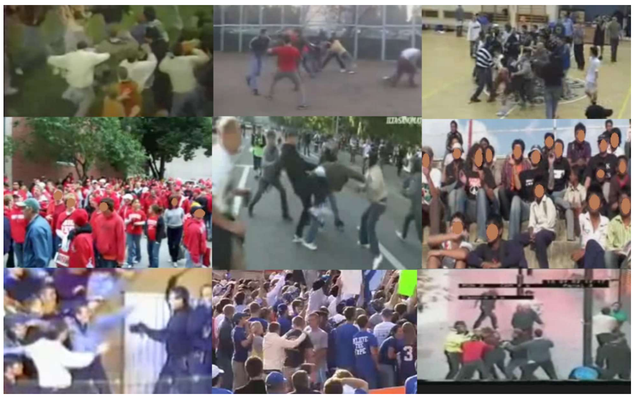

FIGURE 1. Samples from the Violent-Flow dataset [33]

The UCF-101 [34] is an action dataset for recognition and classification collected from YouTube. In total, contains 101 classes of actions and activities, and more than 13,320 videos. For this work, the classes of videos selected were: "punch" for the violence set and "walking the dog" and "biking" for the nonviolence set. Therefore, resulting in 417 training videos and 88 testing videos. Figure 2 presents some examples from the extracted classes for this work.

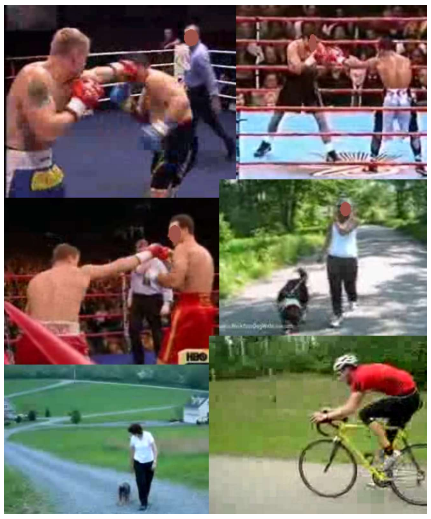

FIGURE 2. Samples from the UCF-101 dataset [34]

The HMDB [35] is a large dataset to recognize actions with 51 classes. Each containing at least 101 videos, for a total of 6,766 videos extracted from digital movies and YouTube videos. For this work were selected the classes "hit", "kick", "punch" and, "shot gun" for the violence set; and "hug", "kiss", "run", "walk" and, "sit" for the nonviolence set. Thus, resulting in 612 training videos and 107 testing videos. Figure 3 shows a few samples from the selected classes of the dataset.

Lastly, the Moments in Time [36] is a research project dedicated to building a dataset for recognizing and understanding video actions. Currently, the dataset includes a collection of approximately one million 3-second videos, corresponding to 339 different classes, involving people, animals, objects or natural phenomena.

For the violence set the classes used were "aiming", "attacking", "boxing", "destroying", "fighting", "hitting", "kicking", "punching", "shooting" and, "wrestling". For nonviolence were used "bicycling", "celebrating", "exercising", "hugging", "jogging" and "running", "kissing", "siting" and, "walking".

Due to the diversity of scenes in this dataset, even within the same class, a careful manual selection of videos was necessary, which resulted in 994 training videos and 162 testing videos. In Figure 4 is displayed some frames from the dataset.

The Table 1 shows the contribution of videos from each dataset to the final combined dataset. The dataset, assembled for this work, contains a total of 2670 videos, in which 2269 videos belongs to the training set and 401 to the testing set. From these videos, the violence class has a total of 1193 videos, in which 1014 are assign to the training set and 179 to the testing set. The nonviolence class contains 1477 videos in which 1255 are assign to the training set and 222

[1]https://www.openu.ac.il/home/hassner/data/violentflows/

[2]https://www.crcv.ucf.edu/data/UCF101/

[3]https://serre-lab.clps.brown.edu/resource/hmdb-a-large-human-motion-database/

[4]http://moments.csail.mit.edu/

FIGURE 3. Samples from the HMDB dataset [35]

FIGURE 4. Samples from the Moments in Time dataset [36]

to the testing set. The difference between the number of train and test videos is due to the greater number of nonviolence classes in the action recognition datasets.

TABLE 1. Distribution of videos gathered from public datasets.

Dataset		Violence	Nonviolence	Total
Train	Violent-Flow	123	123	246
	UCF-101	146	271	417
	HMDB	250	362	612
	Moments in Time	495	499	994
Training Set		**1014**	**1255**	**2269**
Test	Violent-Flow	22	22	44
	UCF-101	40	48	88
	HMDB	45	62	107
	Moments in Time	72	90	162
Testing Set		**179**	**222**	**401**
Total		1193	1477	**2670**

The total video length for the training set is 128 minutes, and 22 minutes for the testing set. The average length of videos is 3.35 seconds. The total duration length distribution of videos is presented in Figure 5. It is possible to observe that most of the videos have a duration lower than 3 seconds. This is a common characteristic of violence videos, in which the actions happen very quickly. Instances higher than 4 seconds usually corresponds to the nonviolence set, with actions such as walking and running.

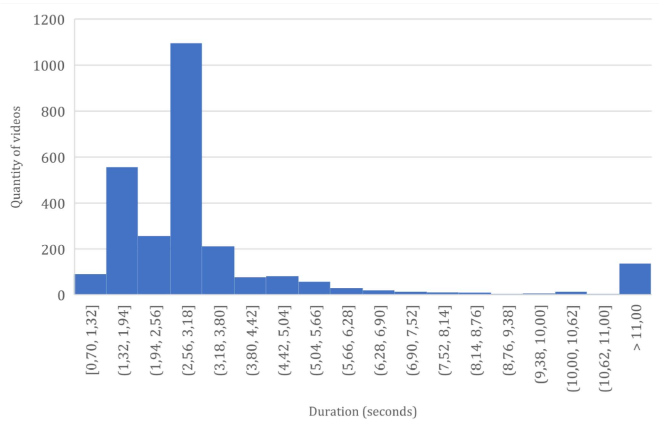

FIGURE 5. Histogram of video duration.

Furthermore, the final dataset provides great diversity in actions, with large variations in camera position, appearance and pose of objects and people, scale, viewpoint, cluttered backgrounds, different lighting conditions, wide variations in motion, video quality, and occlusion. This allows the models to learn and predict the most diverse actions and activities characterized between violent and nonviolent.

IV. PROPOSED METHOD

CNNs have become ubiquitous in Computer Vision since the popularization of the AlexNet architecture [37]. The general trend has been to develop deeper and more complex networks to achieve greater accuracy. However, these improvements do not necessarily make architectures more efficient in terms of number of parameters, model size and processing speed [38]. In many applications, such as robotics, autonomous cars, augmented reality and security, the recognition task needs to be employed at a specific time on a restricted computing platform, due to time constraints or space allocation.

Recently, there are attempts to develop CNN architectures for recognition and detection tasks focused on application in devices with limited computational power, such as smartphones and embedded systems. In this work, it was adapted the most popular of these architectures to the binary classification for violence recognition.

In the following subsections, a description of the used mobile CNN architectures is presented, with a brief introduction about the main method used by them. It is also shown the preprocessing steps, the network hyperparameters and the final number of parameters for each architecture.

A. PREPROCESSING & NETWORK HYPERPARAMETERS

The videos applied on the mobile CNNs were converted into a series of images, with dimensions 227x227x3 or

224x224x3, depending on the CNN architecture. These dimensions are considered ideal for features representation and extraction during training, as large images require high computational power, and in small images, the CNN may not be able to extract significant information. In addition, in-place/on-the-fly data augmentation techniques were used to better generalize the dataset [39]. Thus, each image in a batch is transform by a series of random operations in real time during the training process, among them:

- Mirroring;
- Approximation (up to 20%);
- Rotation (up to 20%);
- Width and height offset (up to 20%).

In addition, this work used the Adam optimization algorithm instead of the traditional SGD optimizer, responsible for minimizing the objective function [40], [41]. Adam's algorithm uses adaptive learning methods to find individual learning rates for each parameter and enable ANNs to train faster. The learning rate lr, which represents the weight update step, has been set at $lr = 1 \cdot 10^{-5}$. Due to memory capabilities, the batch size was set in 32 samples per iteration.

B. MOBILE ARCHITECTURES

A strategy to reduce parameters and operations of mobile CNN architectures is to adapt deep sparse convolutional structures with small filter sizes. That is, instead of the traditional method of adding layer after layer in a sequential order, these architectures add layers in a sequential and parallel combination, normally structured in modules [42].

The final architecture is composed by merging these modules. This method is presented in Figure 6 and an example of a sparsely connected module seen in the popular Inception architecture [43] is shown in Figure 7.

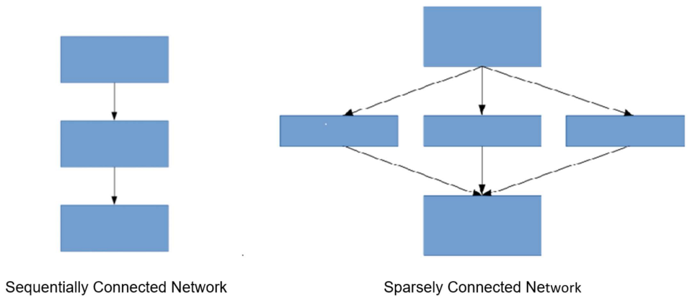

FIGURE 6. Types of neural network connections.

The Inception module introduce a combination of layers, with 1x1, 3x3 and 5x5 convolution layers and max pooling layer sparsely connected. The output from these layers are concatenated into a single output vector, forming an input for the next step [44]. This technique was then expanded in the mobile architectures, mainly because it permits a CNN to capture more details and features in diverse scales, with a reduced number of parameters (compared to a sequentially connected network) [45].

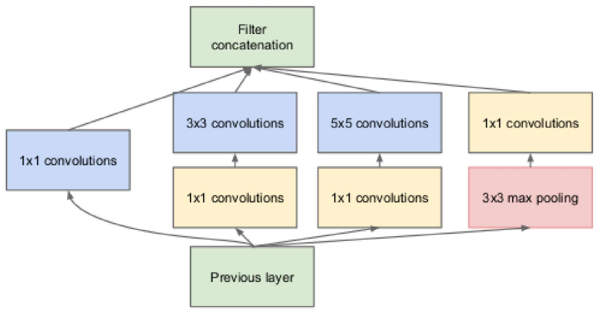

FIGURE 7. The Inception module [43].

1) SqueezeNet

SqueezeNet [46] is a architecture that achieves AlexNet accuracy level [47], yet with fewer parameters. This architecture uses a combination of modules with the idea of squeeze and expand layers. In the squeeze layer there are only 1x1 convolutional filters, whereas in the expand layer there is a combination of 1x1 and 3x3 filters. This high use of 1x1 convolutional filters reduces considerably the number of parameters. After all, 1x1 filters use significant less parameters than larger convolutional filters. For example, a 1x1 filter has 9 times less parameters than a 3x3 convolution filter.

Adapted for this work, the SqueezeNet model presents only 735.94 thousand parameters, which is a number far lower than the AlexNet original architecture, that has 62.3 millions parameters [48]. Figure 8 shows the SqueezeNet module, with the squeeze and expand layers. Also, the architecture uses the ReLU activation function [49].

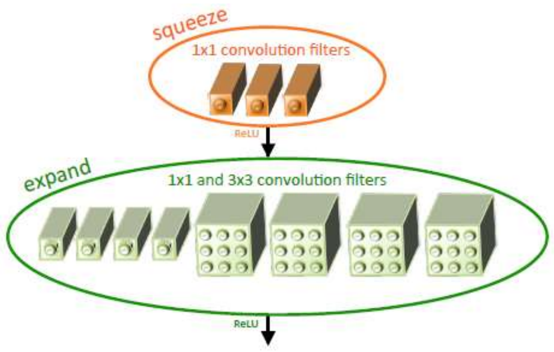

FIGURE 8. The Squeezed module [50]

2) MobileNet-v1 & MobileNet-v2

The MobileNet architecture [51] is primarily based on depthwise separable convolution, in which factors a traditional convolution into a depthwise convolution followed by a pointwise convolution. In other words, first a spatial convolution is performed independently for each channel, then by a 1x1 convolution across all channels. This approach was found to be easier than the normal 3D convolution [52].

Thus, the MobileNet model adapted for this work has a total number of 3.23 million parameters.

In the second version of the MobileNet architecture [53], an inverted residual sparse structure was introduced, consisting of 1x1 convolution, depthwise separable convolution and the use of a linear function. Adapted for this work, the MobileNet-v2 model has 2.26 million parameters, a considerably lower number of parameters than its first version.

Figure 9 shows the MobileNet-v2 module. The module presents a residual cell (has a residual/identity connection introduced by [54]) with stride of 1, and a resizing cell with a stride of 2. From Figure 9, "conv" is a normal convolution, "dwese" is a depthwise separable convolution, "Relu6" is a ReLu activation function with a magnitude limitation, and "Linear" is the use of the linear function.

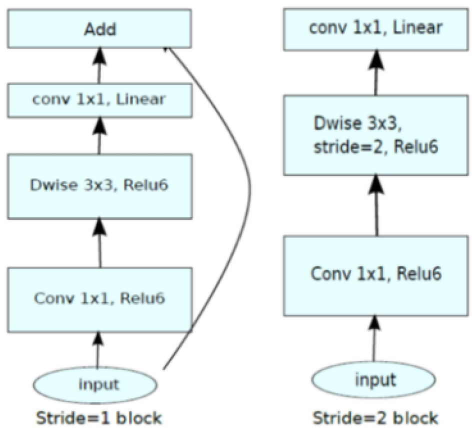

FIGURE 9. The MobileNet-v2 modules [55]

3) NASNet Mobile

The NASNet [56] architecture searches the best combinations of convolution layers, first in a smaller dataset, then expanded the configuration to a lager one. The architecture is composed of a number of sparsely connected convolutions layers, normal and separable, with different filters sizes (1x1, 3x3, 5x5 and 7x7). The authors also developed a mobile version, which was adapted for this work and has 4.27 million parameters.

Figure 10 presents the NASNet module used. The NASNet architecture is composed of normal cells and reduction cells. Normal cells are convolution layers that return feature maps, and reduction cells resize the features maps by a factor. From Figure 10 "sep" means depthwise separable convolutions, "identity" are the residual/identity connections, "avg" are layers of average pooling, and "max" are layers of max pooling.

V. EXPERIMENTS

In this section, computational performance and results are compared and analyzed for each developed model. The error, accuracy and number of parameters were compared, as well as an diagram to verify the effectiveness of the models.

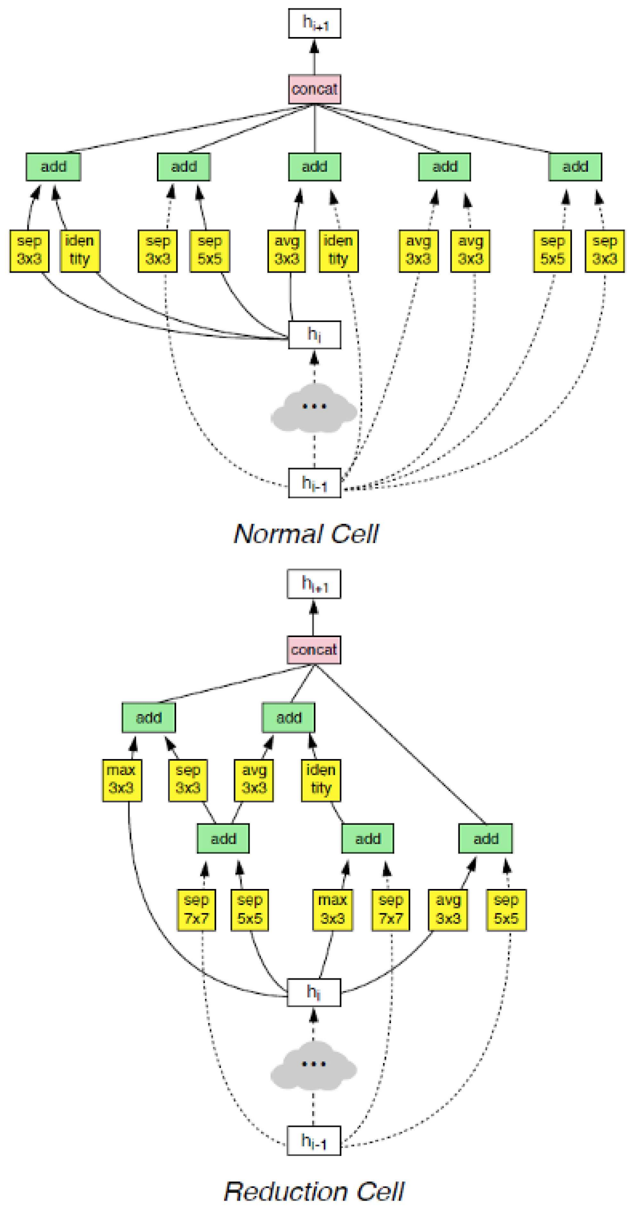

FIGURE 10. The NASNet modules [56].

The Figure 11 displays the accuracy and error per epoch for the training set. The graph shows that the accuracy increases progressively while the error decreases. The MobileNets and NASNet have similar outputs, and the SqueezeNet is slightly less effective. In addition, Figure 12 shows the accuracy and error per epoch for the testing set. Again, the accuracy and the error of the MobileNets and NASNet models are similar, with slightly worse results from SqueezeNet.

It is also noticeable the presence of overfitting soon after the fifth epoch in the error graph from Figure 12. The errors increase as the models have already learned all patterns from the training set, including noise and outliers, and can no

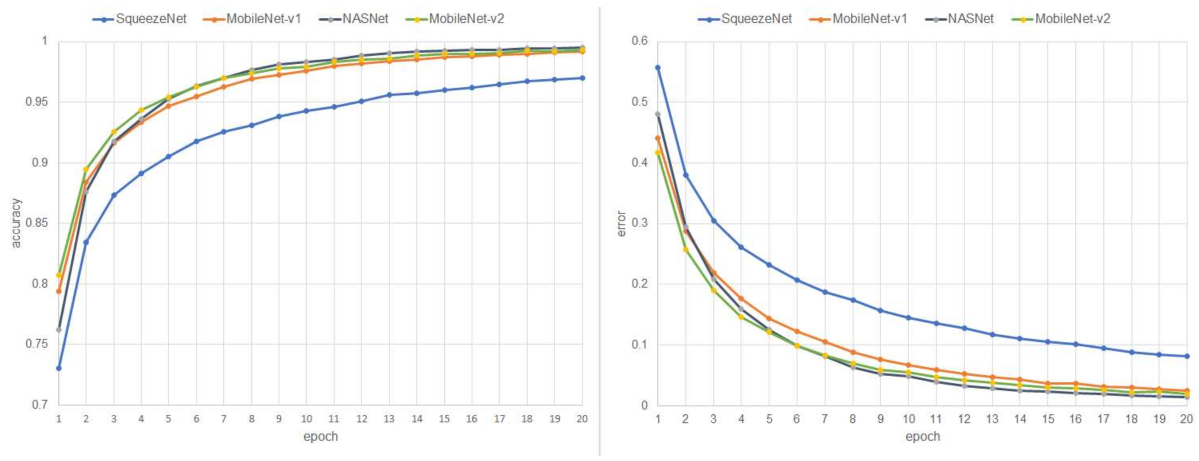

FIGURE 11. Accuracy (left) and error (right) for the training set.

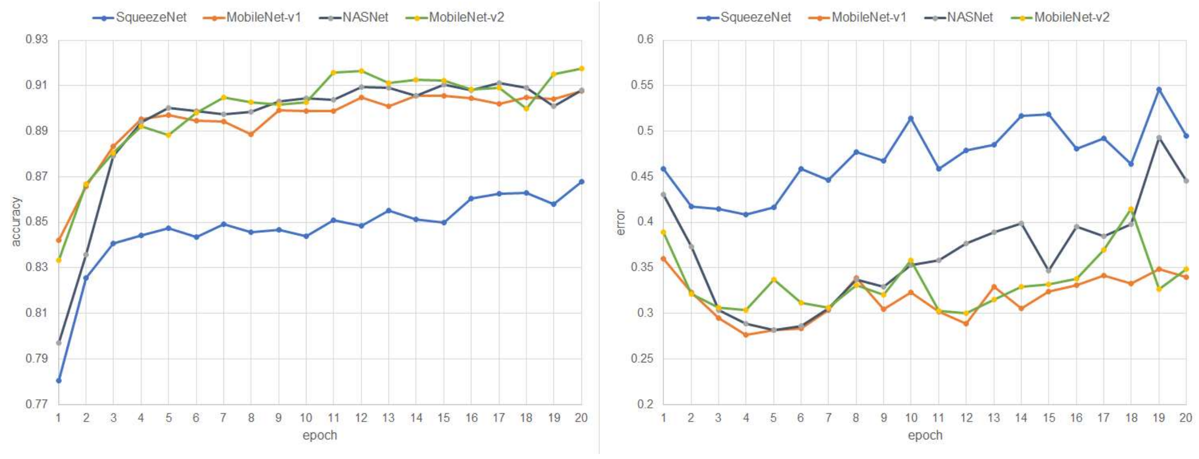

FIGURE 12. Accuracy (left) and error (right) for the testing set.

longer generalize new samples from the testing set. This graph also shows how quickly the models are able to learn all patterns from the dataset and correctly predict new samples.

Therefore, from Figure 12, it is possible to identify in which epoch the best result is obtained, that is, the epoch with the highest accuracy and lowest error. For the SqueezeNet and MobileNet-v1 architectures this happened in the forth epoch. SqueezeNet achieved an accuracy of 0.8702 and an error of 0.3486, and MobileNet-v1 achieved an accuracy of 0.9048 and an error of 0.2888. For the Mobile NASNet it occurred in the fifth epoch, with an accuracy of 0.9001 and an error of 0.2813.

Lastly, for the MobileNet-v2 it happened in twelfth epoch, with an accuracy of 0.9163 and an error of 0.2997. Thus, the best error result belongs to the MobileNet-v1 with an error of 0.2888 and the best accuracy belongs to the MobileNet-v2 with an accuracy of 0.9163. Although, as observed earlier, MobileNet-v1, MobileNet-v2 and NASNet show very close results, SqueezeNet performed slightly worse.

Figure 13 shows the total number of parameters from each adapted architecture for this work. It is possible to observe that the model with the lowest number of parameters belongs to the SqueezeNet with only 735.94 thousand parameters, three times lower than the second model with the lowest number of parameters, MobileNet-v2 with 2.26 million parameters. The MobileNet-v1 and NASNet have 3.23 and 4.27 million parameters, respectively.

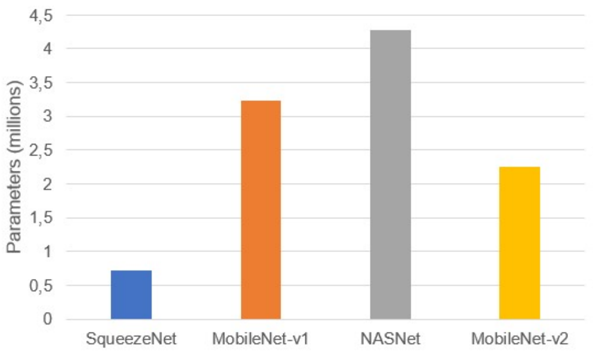

FIGURE 13. Comparative total number of parameters for each architecture.

Lastly, Figure 14 shows the accuracy versus the total number of parameters for each adapted architecture. The SqueezeNet have the lowest number of parameters from all the models, however have a lower accuracy rate. The MobileNet-v1, MobileNet-v2 and the NASNet shows high accuracy, but demand more processing power. Therefore, analyzing the graphic, it is possible to observe that the MobileNet-v2 presents the best cost-benefit as it has a high accuracy while keeping a low number of parameters. However, limited systems will still be able to run the SqueezeNet model more effectively, due to its reduced number of parameters.

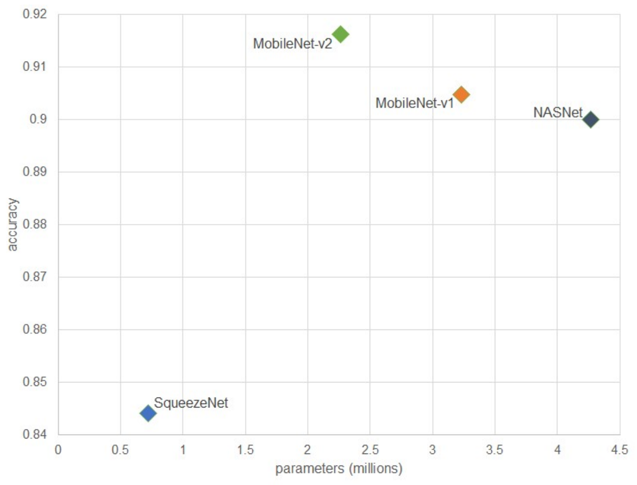

FIGURE 14. Scatter plot of accuracy vs. number of parameters.

VI. PROTOTYPE

SqueezeNet, MobileNet-v1, MobileNet-v2 and NASNet Mobile are architectures specially developed focused on mobile and embedded applications. Therefore, to prove if they are truly able to run a mobile application for complex tasks, such as violent recognition, the Raspberry Pi 4 embedded platform was selected as a prototype. Table 2 presents the development environment. The framework used for both Laptop with GPU and Raspberry Pi 4 was TensorFlow 2.6.0. Moreover, for the Laptop with GPU, was used CUDA 11.4 and cuDNN 8.2.4.

TABLE 2. Development environment of Raspberry Pi 4 and desktop.

	Raspberry Pi 4	Laptop
OS Distributor	Debian	Ubuntu
Description	Debian GNU/Linux 11 (bullseye)	Ubuntu 18.04.6 LTS (Bionic Beaver)
Release	11	18.04
Kernel Version	5.10.92-v8+	5.4.0-84-generic

This section presents the operation process of the prototype, with experiments on the average runtime and the FPS (frames-per-second) that the prototype is able to reach while running the models. Comparative tables of the same models running on a more complex platform are also presented.

The prototype for this work is able to run a CNN model, read and transform a video signal from a file or a USB webcam to an input signal, that allows prediction to be made by the model. The result of this prediction is then displayed over the screen along with the respective label. The transformations necessary to feed the CNN model consist of transforming the video signal into a series of images, resizing and normalizing the images. The Figure 15 presents an example of the prototype operation process and the Figure 16 shows the prototype running a model.

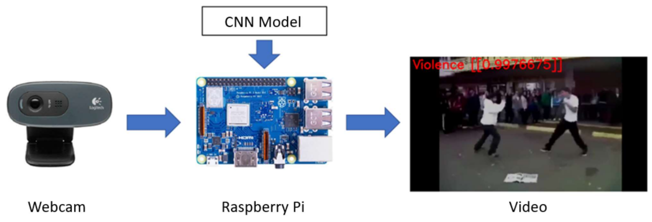

FIGURE 15. The prototype operation process.

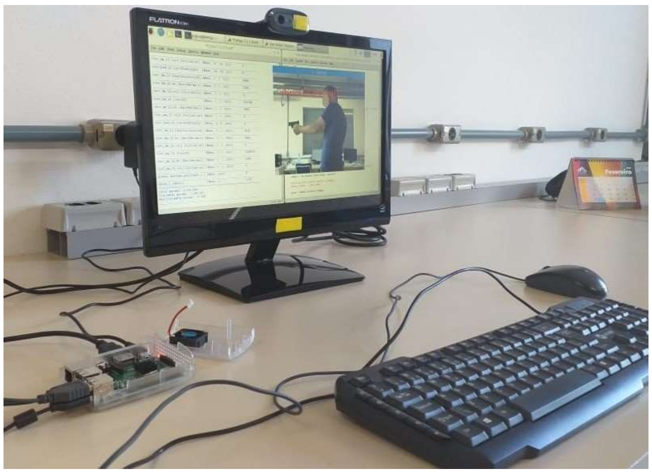

FIGURE 16. The prototype running a model.

After the attempt to run the developed mobile models, Table 3 shows the adapted architecture, with its total number of parameters, the average runtime and the average FPS rate. These values are obtained based on the average value to predict 30 seconds of life streaming with 10 repetitions.

TABLE 3. Comparative number of parameters, average runtime and FPS on Raspberry Pi 4.

Architecture	Parameters	Runtime (s)	FPS
SqueezeNet	735,937	98.25	4.19
MobileNet-v1	3,229,889	113.60	3.74
MobileNet-v2	2,259,265	103.05	4.05
NASNet	4,270,773	153.86	3.02

From Table 3, it is possible to observe that the SqueezeNet architecture has the shortest average runtime. This result is expected, as the SqueezeNet architecture has the smallest number of parameters and makes high use of small convolutional filters as a technique for a lower computational cost.

However, the MobileNet-v1 and MobileNet-v2 architectures have a close average runtime, even though they have a higher number of parameters (compared to non-mobile CNNs). Nevertheless, the MobileNet architectures are much deeper, with many normal and depth-separable convolution layers, capable of extracting more features and information, and presenting superior accuracy.

The Raspberry Pi 4 platform was not efficient to run the NASNet Mobile architecture. This is due to NASNet is an extremely complex architecture, with several layers sparsely connected and various filter sizes, which makes it difficult to run even on more complex platforms.

To compare the influence of a more powerful setting, the same models were executed on a laptop with a dedicated GPU with the following specifications: Core i7 7700HQ processor, 16GB RAM, NVIDIA GTX 1050Ti GPU. The results are presented in Table 4.

TABLE 4. Comparative number of parameters, average runtime and FPS on desktop.

Architecture	Parameters	Runtime (s)	FPS
SqueezeNet	735,937	1.46	24.22
MobileNet-v1	3,229,889	1.32	20.43
MobileNet-v2	2,259,265	1.32	22.76
NASNet	4,270,773	2.24	14.70

According to Table 4, the influence of more powerful setting with a dedicated GPU is clear. Models running on the laptop perform on average 98% better than those running on the Raspberry Pi platform. The use of GPUs improves performance due to parallel operations, where the multiple cores available in GPUs allow running multiple operations at the same time.

VII. WARNING SYSTEM

The work of violence recognition is typically a binary classification problem, that is, it has only two classes: violence and nonviolence. In these cases, the function for calculating the probability of belonging to a class is performed by the sigmoid function. The output values can only assume values between 0 and 1. Therefore, a threshold is established to define to which class an input belongs. For comparison purposes, Table 5 presents the classification results using different algorithms. In this case, the runtime is the average time to predict 1024 images with batch size of 32 and 10 repetitions.

In this application it can be observed that the results of accuracy and F1-score make the application possible, highlighting MobileNet-v2 that obtained a result of 92% accuracy and F1-score of 0.92 for the classification of situations of violence. Table 6 presents a benchmarking with well-established models based on deep neural networks. As can be seen, the use of models that require greater computational effort does not significantly improve the accuracy results in this application. In presenting the comparison between the models in Table 5 and Table 6 the time considered is the total time, which comprises the sum of the training time plus the execution time on the embedded system.

TABLE 5. Comparison of algorithms.

Model	Accur.	Prec.	Recall	F1-score	Time (s)
SqueezeNet	0.87	0.87	0.87	0.87	114.79
MobileNet-v1	0.90	0.91	0.90	0.90	114.92
MobileNet-v2	0.92	0.92	0.91	0.92	104.37
NASNetMobile	0.90	0.90	0.90	0.90	156.28

TABLE 6. Convolutional neural network benchmark.

Model	Accur.	Prec.	Recall	F1-score	Time (s)
VGG-16	0.91	0.92	0.91	0.91	1,276.85
Inception V3	0.91	0.92	0.90	0.91	262.26
ResNet50	0.92	0.92	0.92	0.92	483.2

After training the model, the outputs generated by the sigmoid function was carefully analyzed frame by frame. It was observed that the outputs close to the threshold correspond to cases in which actions and activities have common characteristics, such as the position of the bodies or the arms around someone. For example, nonviolence scenes that feature actions of "hugging" and "kissing" show many characteristics in common to violence scenes that feature actions of "fighting", "attacking", and "beating". Precisely, it is in these cases that the algorithm can perform the prediction incorrectly.

To reduce this problem, a "third class" was created in the sigmoid function, that is, instead of using only a threshold, a zone of uncertainty was delimited. Thus, in cases where the algorithm is unable to correctly classify an input, the vigilant operator is alerted that the scene may or may not contain violence. The uncertainty zone was defined between the limits $[0.35 - 0.65]$ as presented in Figure 17.

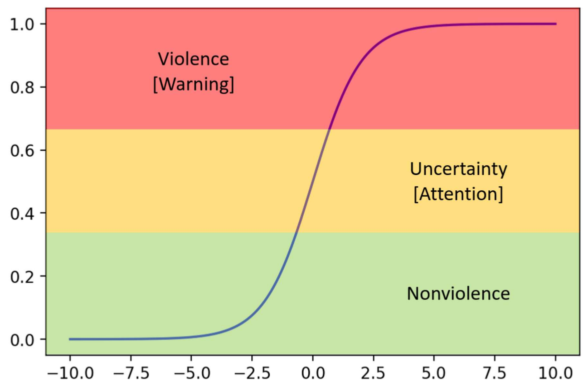

FIGURE 17. Warning system limits for the sigmoid function.

Another important observation is that, in some cases, the frames right before acts of violence, usually with quick and expressive actions and fast movement gestures, the algorithm tends to generate a prediction in the uncertainty zone. Thus, in cases where there is a gradual evolution of violence in the behavior of those involved, such as sudden movements and accusations to finally acts of physical attacks, the algorithm is able to draw the attention of the vigilant operator to be aware of a possible occurrence and take appropriate preventive action.

Figure 18 shows the output generated by the SqueezeNet model, executed on the Raspberry Pi embedded platform, where the label and the prediction result is displayed on the screen. It possible to observe that nonviolence actions have a prediction result close to zero, violent actions have a prediction result close to one and cases of pre-fights, with quick movements expressing violence, have a prediction result around 0.5.

Moreover, Figure 19 and Figure 20 display the model output applied in crowded and non-crowded environments. The image samples were taken from the testing set, and show that the trained models are able to successfully recognize violence in both environments with diverse backgrounds.

VIII. CONCLUSION

This work evaluated how mobile CNNs can perform the task of automatic violence recognition in a new dataset of 2670 videos containing scenes of violence in crowded and non-crowded environments, collected from various public datasets. The experiments have shown that high classification accuracy can be achieved using mobile architectures with a lower number of parameters and, it was able to achieve up to 92.05% of accuracy.

A low-cost prototype of an intelligent monitoring system was presented on a Raspberry Pi embedded platform and used to compare the performance of different developed mobile CNN models, which was able to run a real-time model on up to 4.19 FPS. Experimental results demonstrated that it is possible to achieve with mobile CNN models even on platforms with limited processing power, proving a higher efficiency of the models without high deployment costs.

After careful analysis of the models output, it was noticed that the incorrect prediction of the models occurs when actions present similarities between the two classes of violence and non-violence. For example, hugging can have many characteristics in common with violent actions when the video is analyzed frame by frame. To try to minimize this behavior, a third class was developed on the response of models with warning function or safety monitors in cases where the algorithm is not able to predict correctly. Through this system, in some cases prior to acts of violence, usually with actions and gestures of fast and expressive movements, the algorithm is used to generate a response in this attention zone. Thus, when there is a gradual evolution of violence in the behavior of those involved, the algorithm is able to call the attention of monitoring agents in order to prevent occurrences.

As future work, we can focus on a study of the impact of using mobile CNN architectures on more powerful embedded platforms. Moreover, we aim to improve the dataset, with videos of actions of violence and nonviolence, above all in public places such as malls, airports, subways, parks, and sports stadiums, in order to improve the applicability and accuracy of the models. Another possibility is the detection and location of the violent occurrences in the videos. This could be accomplished by using BoVW (Bag of Visual Words) or by more advanced segmentation techniques.

ACKNOWLEDGMENTS

The authors would like to thank the Coordination for the Improvement of Higher Education Personnel (CAPES- Brazil), the PIBIC/FURB Program and the University of Blumenau (FURB) for their support in this work.

This work was supported by national funds through the Fundação para a Ciência e a Tecnologia, I.P. (Portuguese Foundation for Science and Technology) by the project UIDB/05064/2020 (VALORIZA—Research Centre for Endogenous Resource Valorization) and it was partially supported by Fundação para a Ciência e a Tecnologia under Project UIDB/04111/2020, and ILIND–Instituto Lusófono de Investigação e Desenvolvimento, under roject COFAC/ILIND/COPELABS/3/2020.

REFERENCES

[1] 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. 203668, 2018.

[2] E. Bermejo Nievas, O. Deniz Suarez, G. Bueno García, and R. Sukthankar, “Violence detection in video using computer vision techniques,” in Computer Analysis of Images and Patterns, vol. 6855, (Berlin, Germany), pp. 332–339, Springer, 2011.

[3] A. B. Mabrouk and E. Zagrouba, “Abnormal behavior recognition for intelligent video surveillance systems: A review,” Expert Systems with Applications, vol. 91, pp. 480–491, 2018.

[4] S. Vishwakarma and A. Agrawal, “A survey on activity recognition and behavior understanding in video surveillance,” The Visual Computer, vol. 29, no. 10, pp. 983–1009, 2013.

[5] O. P. Popoola and K. Wang, “Video-based abnormal human behavior recognition: A review,” IEEE Transactions on Systems, Man, and Cybernetics, vol. 42, no. 6, pp. 865–878, 2012.

[6] M. Rusci, D. Rossi, E. Farella, and L. Benini, “A sub-mw iot-endnode for always-on visual monitoring and smart triggering,” IEEE Internet of Things Journal, vol. 4, no. 5, pp. 1284–1295, 2017.

[7] S. F. Stefenon, C. Kasburg, R. Z. Freire, F. C. Silva Ferreira, D. W. Bertol, and A. Nied, “Photovoltaic power forecasting using wavelet neuro-fuzzy for active solar trackers,” Journal of Intelligent & Fuzzy Systems, vol. 40, no. 1, pp. 1083–1096, 2021.

[8] G. Cerutti, R. Prasad, A. Brutti, and E. Farella, “Compact recurrent neural networks for acoustic event detection on low-energy low-complexity platforms,” IEEE Journal of Selected Topics in Signal Processing, vol. 14, no. 4, pp. 654–664, 2020.

[9] A. Cimatti and S. Tonetta, “Contracts-refinement proof system for component-based embedded systems,” Science of Computer Programming, vol. 97, pp. 333–348, 2015.

[10] A. Cimatti, S. Mover, and S. Tonetta, “Quantifier-free encoding of invariants for hybrid systems,” Formal Methods in System Design, vol. 45, no. 2, pp. 165–188, 2014.

[11] F. Montagna, M. Buiatti, S. Benatti, D. Rossi, E. Farella, and L. Benini, “A machine learning approach for automated wide-range frequency tagging analysis in embedded neuromonitoring systems,” Methods, vol. 129, pp. 96–107, 2017.

[12] S. F. Stefenon, C. Kasburg, A. Nied, A. C. R. Klaar, F. C. S. Ferreira, and N. W. Branco, “Hybrid deep learning for power generation forecasting in active solar trackers,” IET Generation, Transmission & Distribution, vol. 14, no. 23, pp. 5667–5674, 2020.

[13] M. Rusci, D. Rossi, M. Lecca, M. Gottardi, E. Farella, and L. Benini, “An event-driven ultra-low-power smart visual sensor,” IEEE Sensors Journal, vol. 16, no. 13, pp. 5344–5353, 2016.

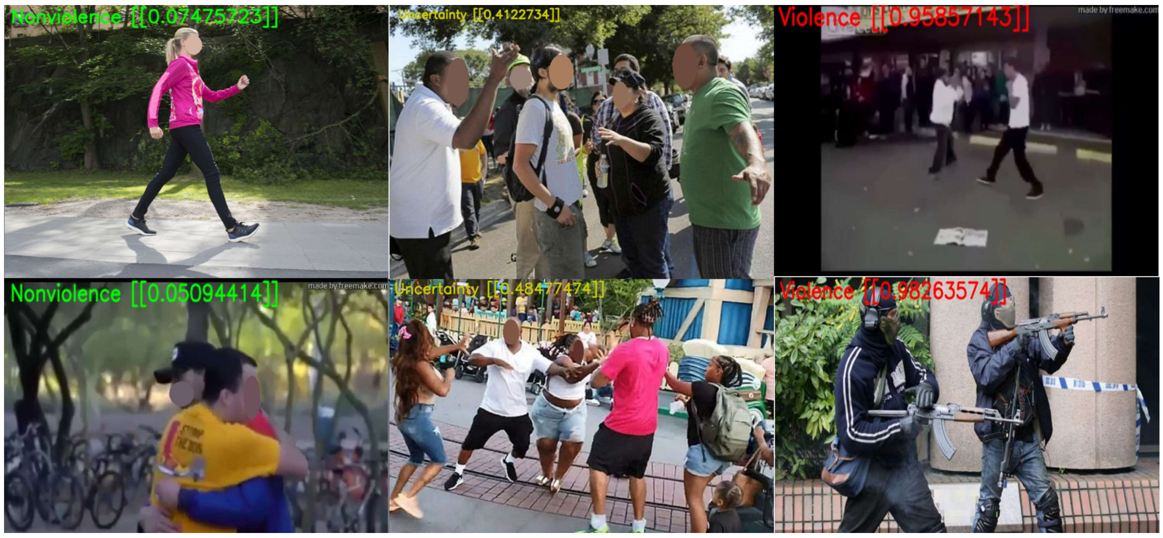
Nonviolence [[0.07475723]]
Uncertainty [[0.4122734]]
Violence [[0.95857143]]
made by freemake.com
Nonviolence [[0.05094414]]
made by freemake.com
Uncertainty [[0.48477474]]
Violence [[0.98263574]]

FIGURE 18. Model output.

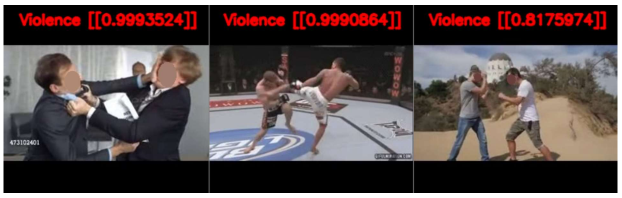

FIGURE 19. Model output of violence in non-crowded environments.

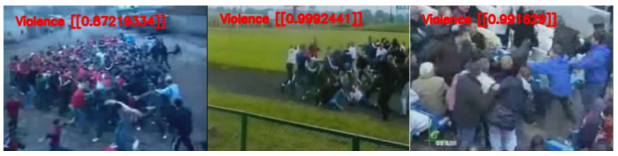

FIGURE 20. Model output of violence in crowded environments.

[14] N. F. Sopelsa Neto, S. F. Stefenon, L. H. Meyer, R. Bruns, A. Nied, L. O. Seman, G. V. Gonzalez, V. R. Q. Leithardt, and K.-C. Yow, "A study of multilayer perceptron networks applied to classification of ceramic insulators using ultrasound," Applied Sciences, vol. 11, no. 4, p. 1592, 2021.
[15] S. F. Stefenon, M. P. Corso, A. Nied, F. L. Perez, K.-C. Yow, G. V. Gonzalez, and V. R. Q. Leithardt, "Classification of insulators using neural network based on computer vision," IET Generation, Transmission & Distribution, *Preprint*, pp. 1–12, 2022.
[16] D. Liu, H. Zhang, and P. Zhou, "Video-based facial expression recognition using graph convolutional networks," in International Conference on Pattern Recognition, vol. 25, (Milan, Italy), pp. 607–614, IEEE, 2021.
[17] N. Varshney and B. Bakariya, "Deep convolutional neural model for human activities recognition in a sequence of video by combining multiple cnn streams," Multimedia Tools and Applications, *Preprint*, pp. 1–13, 2021.
[18] G. H. dos Santos, L. O. Seman, E. A. Bezerra, V. R. Q. Leithardt, A. S. Mendes, and S. F. Stefenon, "Static attitude determination using convolutional neural networks," Sensors, vol. 21, no. 19, p. 6419, 2021.
[19] M. P. Corso, F. L. Perez, S. F. Stefenon, K.-C. Yow, R. García Ovejero, and V. R. Q. Leithardt, "Classification of contaminated insulators using k-nearest neighbors based on computer vision," Computers, vol. 10, no. 9, p. 112, 2021.
[20] J. Li, X. Jiang, T. Sun, and K. Xu, "Efficient violence detection using 3d convolutional neural networks," in International Conference on Advanced Video and Signal Based Surveillance, vol. 16, (Taipei, Taiwan), pp. 1–8, IEEE, 2019.
[21] Q. Dai, R.-W. Zhao, Z. Wu, X. Wang, Z. Gu, W. Wu, and Y.-G. Jiang, "Fudan-huawei at mediaeval 2015: Detecting violent scenes and affective impact in movies with deep learning," in Multimedia Benchmark Workshop, (Wurzen, Germany), MediaEval, 2015.
[22] S. F. Stefenon, R. Z. Freire, L. H. Meyer, M. P. Corso, A. Sartori, A. Nied, A. C. R. Klaar, and K.-C. Yow, "Fault detection in insulators based on ultrasonic signal processing using a hybrid deep learning technique," IET Science, Measurement & Technology, vol. 14, no. 10, pp. 953–961, 2021.
[23] F. Fernandes, S. F. Stefenon, L. O. Seman, A. Nied, F. C. S. Ferreira, M. C. M. Subtil, A. C. R. Klaar, and V. R. Q. Leithardt, "Long short-term memory stacking model to predict the number of cases and deaths caused by covid-19," Journal of Intelligent & Fuzzy Systems, *Preprint*, pp. 1–14, 2021.
[24] C. Ding, S. Fan, M. Zhu, W. Feng, and B. Jia, "Violence detection in video by using 3d convolutional neural networks," in Advances in Visual Computing, vol. 8888, (Cham, Germany), pp. 551–558, Springer, 2014.
[25] P. Wang, P. Wang, and E. Fan, "Violence detection and face recognition based on deep learning," Pattern Recognition Letters, vol. 142, pp. 20–24, 2021.
[26] M. Sabokrou, M. Fayyaz, M. Fathy, Z. Moayed, and R. Klette, "Deep-anomaly: Fully convolutional neural network for fast anomaly detection in crowded scenes," Computer Vision and Image Understanding, vol. 172, pp. 88–97, 2018.
[27] M. H. Gauswami and K. R. Trivedi, "Implementation of machine learning for gender detection using cnn on raspberry pi platform," in International Conference on Inventive Systems and Control, vol. 2, (Coimbatore, India), pp. 608–613, IEEE, 2018.
[28] P. Suja, S. Tripathi, et al., "Real-time emotion recognition from facial images using raspberry pi ii," in International Conference on Signal Processing and Integrated Networks, vol. 3, (Noida, India), pp. 666–670, IEEE, 2016.
[29] A. Medeiros, A. Sartori, S. F. Stefenon, L. H. Meyer, and A. Nied, "Comparison of artificial intelligence techniques to failure prediction in contaminated insulators based on leakage current," Journal of Intelligent & Fuzzy Systems, *Preprint*, pp. 1–14, 2022.
[30] V. Mazzia, A. Khaliq, F. Salvetti, and M. Chiaberge, "Real-time apple detection system using embedded systems with hardware accelerators: An edge ai application," IEEE Access, vol. 8, pp. 9102–9114, 2020.
[31] V. Leithardt, D. Santos, L. Silva, F. Viel, C. Zeferino, and J. Silva, "A solution for dynamic management of user profiles in iot environments," IEEE Latin America Transactions, vol. 18, no. 07, pp. 1193–1199, 2020.
[32] M. Dua, R. Singla, S. Raj, A. Jangra, et al., "Deep cnn models-based ensemble approach to driver drowsiness detection," Neural Computing and Applications, vol. 33, no. 8, pp. 3155–3168, 2021.

[33] T. Hassner, Y. Itcher, and O. Kliper-Gross, “Violent flows: Real-time detection of violent crowd behavior,” in Computer Society Conference on Computer Vision and Pattern Recognition Workshops, vol. 1, (Providence, USA), pp. 1–6, IEEE, 2012.
[34] K. Soomro, A. R. Zamir, and M. Shah, “Ucf101: A dataset of 101 human actions classes from videos in the wild,” arXiv preprint arXiv:1212.0402, 2012.
[35] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, “Hmdb: a large video database for human motion recognition,” in International Conference on Computer Vision, (Barcelona, Spain), pp. 2556–2563, IEEE, 2011.
[36] M. Monfort, A. Andonian, B. Zhou, K. Ramakrishnan, S. A. Bargal, T. Yan, L. Brown, Q. Fan, D. Gutfreund, C. Vondrick, and A. Oliva, “Moments in time dataset: One million videos for event understanding,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 42, no. 2, pp. 502–508, 2020.
[37] H.-J. Yoo, “Deep convolution neural networks in computer vision: a review,” IEIE Transactions on Smart Processing & Computing, vol. 4, no. 1, pp. 35–43, 2015.
[38] P. R. R. De Souza, K. J. Matteussi, A. D. S. Veith, B. F. Zanchetta, V. R. Leithardt, A. L. Murciego, E. P. De Freitas, J. C. Dos Anjos, and C. F. Geyer, “Boosting big data streaming applications in clouds with burstflow,” IEEE Access, vol. 8, pp. 219124–219136, 2020.
[39] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, “Understanding data augmentation for classification: when to warp?,” in international conference on digital image computing: techniques and applications, vol. 1, (Gold Coast, Australia), pp. 1–6, IEEE, 2016.
[40] S. F. Stefenon, L. O. Seman, N. F. S. Neto, L. H. Meyer, A. Nied, and K.-C. Yow, “Echo state network applied for classification of medium voltage insulators,” International Journal of Electrical Power & Energy Systems, vol. 134, p. 107336, 2022.
[41] S. F. Stefenon, M. H. D. M. Ribeiro, A. Nied, K.-C. Yow, V. C. Mariani, L. dos Santos Coelho, and L. O. Seman, “Time series forecasting using ensemble learning methods for emergency prevention in hydroelectric power plants with dam,” Electric Power Systems Research, vol. 202, p. 107584, 2022.
[42] D. Rajkumar, “Image classification using network inception-architecture & appications,” International Journal of Innovative Research in Science Engineering and Technology, vol. 10, no. 1, pp. 329–333, 2021.
[43] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in Conference on Computer Vision and Pattern Recognition, (Boston, USA), pp. 1–9, IEEE, 2015.
[44] C. Lin, G. Zhao, Z. Yang, A. Yin, X. Wang, L. Guo, H. Chen, Z. Ma, L. Zhao, H. Luo, et al., “Cir-net: Automatic classification of human chromosome based on inception-resnet architecture,” IEEE/ACM Transactions on Computational Biology and Bioinformatics, *Preprint*, pp. 1–9, 2020.
[45] M. G. D. Dionson and P. B. El Jireh, “Inception-v3 architecture in dermatoglyphics-based temperament classification,” Philippine Social Science Journal, vol. 3, no. 2, pp. 173–174, 2020.
[46] F. Ucar and D. Korkmaz, “Covidiagnosis-net: Deep bayes-squeezenet based diagnosis of the coronavirus disease 2019 (covid-19) from x-ray images,” Medical hypotheses, vol. 140, p. 109761, 2020.
[47] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in Neural Information Processing Systems, vol. 25, pp. 1–9, Curran Associates, 2012.
[48] Y. Xu, G. Yang, J. Luo, and J. He, “An electronic component recognition algorithm based on deep learning with a faster squeezenet,” Mathematical Problems in Engineering, vol. 2020, no. 2940286, pp. 1–11, 2020.
[49] S. F. Stefenon, L. O. Seman, C. S. Furtado Neto, A. Nied, D. M. Seganfredo, F. G. da Luz, P. H. Sabino, J. T. González, and V. R. Q. Leithardt, “Electric field evaluation using the finite element method and proxy models for the design of stator slots in a permanent magnet synchronous motor,” Electronics, vol. 9, no. 11, p. 1975, 2020.
[50] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size,” arXiv preprint arXiv:1602.07360, 2016.
[51] P. N. Srinivasu, J. G. SivaSai, M. F. Ijaz, A. K. Bhoi, W. Kim, and J. J. Kang, “Classification of skin disease using deep learning neural networks with mobilenet v2 and lstm,” Sensors, vol. 21, no. 8, p. 2852, 2021.
[52] S. Phiphiphatphaisit and O. Surinta, “Food image classification with improved mobilenet architecture and data augmentation,” in International Conference on Information Science and System, vol. 3, (Cambridge, UK), pp. 51–56, 2020.
[53] U. Kulkarni, S. Meena, S. V. Gurlahosur, and G. Bhogar, “Quantization friendly mobilenet (QF-MobileNet) architecture for vision based applications on embedded platforms,” Neural Networks, vol. 136, pp. 28–39, 2021.
[54] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Conference on Computer Vision and Pattern Recognition, (Las Vegas, USA), pp. 770–778, IEEE, 2016.
[55] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” in Conference on Computer Vision and Pattern Recognition, (Salt Lake City, USA), pp. 4510–4520, IEEE, 2018.
[56] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, “Learning transferable architectures for scalable image recognition,” in Conference on Computer Vision and Pattern Recognition, (Salt Lake City, USA), pp. 8697–8710, IEEE, 2018.